ESCUELA POLITÉCNICA SUPERIOR
DEPARTAMENTO DE INGENIERÍA
DE SISTEMAS Y AUTOMÁTICA

TRABAJO DE FIN DE GRADO
GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROGRAMACIÓN EN ANDROID ACCESIBLE PARA EL CONTROL DE PRESSMATIC

Madrid, Septiembre 2014

Autor: Alejandro Vega Gómez

Tutor: Alberto Jardón Huete

La discapacidad no es otra cosa que nuestra incapacidad para entender que todos tenemos capacidades diferentes.

Agradecimientos

A mi familia, por todos sus ánimos y apoyo a lo largo de estos años.

A Verónica, porque sin su paciencia esto hubiera sido mucho más difícil.

A mis amigos, por creer en mí.

A la universidad, y más concretamente a los profesores, por haberme formado como ingeniero y más como persona.

Resumen

Este proyecto describe el desarrollo de una aplicación accesible para dispositivos móviles creada para controlar un aparato denominado PRESSMATIC, ideado previamente por Gabriel Barroso de María. PRESSMATIC está destinado principalmente a asistir a personas cuya destreza manual se ha visto reducida. Se trata de un dispositivo electromecánico portátil capaz de producir movimientos automáticos de apertura y cierre, provisto de una pantalla que permite su control. Dicho dispositivo emplea cabezales intercambiables, como pueden ser unas tijeras o unas pinzas, para desempeñar la función que tienen los dedos pulgar e índice.

La aplicación móvil diseñada es capaz de comunicarse con PRESSMATIC mediante las funciones de conectividad bluetooth que ofrecen los dispositivos móviles actuales. Después de establecer la conexión con PRESSMATIC, el usuario se hallará en disposición de utilizar de manera funcional y sencilla las distintas herramientas del dispositivo, accediendo a sus distintos modos de operación. De esta manera es posible manejar el dispositivo desde la aplicación móvil, presentándose como una alternativa sólida al control desde la pantalla existente en PRESSMATIC, aportando una serie de ventajas sobre ésta.

La accesibilidad es un rasgo esencial de la aplicación y, en relación a la misma, se lleva a cabo un estudio de los criterios de accesibilidad existentes, y se analiza la adaptación de dichos criterios en la plataforma escogida (Android). A partir de entonces se inicia el diseño y la implementación de una aplicación con el deseo que pueda ser utilizada por todos y llegue a la máxima cantidad de usuarios posible.

Es digno de mencionar que este proyecto surge a partir del diseño planteado por Gabriel y, a partir de éste, surgen otros proyectos de fin de grado aparte del que se presenta ahora. Hay dos proyectos que han progresado de manera simultánea a éste. Por un lado se encuentra el proyecto de Alonso Rosado, destinado a implementar un protocolo bluetooth estable para la comunicación con PRESSMATIC. Con su colaboración, se ha podido integrar dicho protocolo en la aplicación. Por otro lado, el proyecto de Rodrigo Martín tiene como objetivo crear un control efectivo de PRESSMATIC valiéndose de la pantalla integrada en el mismo. A la hora de diseñar las interfaces del dispositivo móvil y del dispositivo PRESSMATIC, la coordinación de este proyecto y del proyecto actual ha sido crucial para proporcionar una mejor experiencia de usuario y facilitar su uso posterior.

Abstract

This project describes the development of an accessible application for mobile devices created to control a device called PRESSMATIC, invented by Gabriel Barroso de María. PRESSMATIC is intended for assist people whose hands lack of some mobility. It is a portable electromechanical device capable of producing automatic opening and closing movements, provided of a screen that provides its control. This device employs interchangeable tools, such as scissors, to perform the role of the thumb and index fingers.

The mobile application designed is able to communicate with PRESSMATIC by means of the connectivity functions that the actual mobile devices supply. After establishing a connection with PRESSMATIC, the users will have at their disposal the different tools by accessing the different modes of operation. In this way it is possible to control the device from the mobile application, being a solid alternative to the screen existing in PRESSMATIC, offering several advantages.

Accessibility is an essential feature of the application and, related to it, a study on current accessibility criterion is performed and the adaptation of these criterions is analyzed. From then on the design and the implementation of the application are performed in such a way that everybody can benefit from it.

It is worthy of mention that this project arises from the design carried out by Gabriel and, from this project, other final degree projects arise apart from the actual one. There are two projects that have been developed at the same time as the current project. On one hand, the project created by Alonso Rosado is intended for integrate a stable bluetooth protocol to communicate with PRESSMATIC. With his collaboration, it has been possible to integrate it in the application. On the other hand, the objective of Rodrigo´s project is to create an effective control over PRESSMATIC by using the screen provided. When designing the user interfaces of the mobile device and the PRESSMATIC device, the coordination is a key point in order to provide a better user experience.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este apartado se exponen las razones que han llevado al desarrollo del proyecto y se presenta se manera global el mismo, introduciendo las ideas de desarrollo iniciales. Por otro lado, se puntualizan los objetivos a cumplir y las fases que se seguirán para conseguir dichos objetivos.

1.1. Motivación del proyecto

Las bases sobre las que se asienta este proyecto encuentran su origen en una tesis previa realizada por Gabriel Barroso de María bajo el título "Dispositivo automático de apoyo para uso de herramientas requeridas de movimiento manual de pinzado".

En la tesis de Gabriel se expone que existen millones de personas en el mundo que experimentan algún tipo de diversidad funcional y que, entre ellas, se encuentran aquellas que han perdido cierta gracilidad manual. Dadas estas circunstancias, se considera la posibilidad de hacer frente a este problema mediante el diseño de un dispositivo electromecánico, cuyo objetivo es ayudar a personas que han perdido dichas facultades.

El dispositivo en cuestión tiene como objetivo posibilitar a estas personas la realización de tareas o el uso de objetos de la vida cotidiana que requieran dicha destreza manual. Éstas pueden ser tales como agarrar objetos o utilizar herramientas de corte como tijeras o cortaúñas. Dichas acciones, aparentemente sencillas, se convierten en tareas imposibles o muy difíciles debido a la mencionada falta de movilidad.

En la mencionada tesis se lleva a cabo un estudio del estado del arte de este tipo de dispositivos para conocer con seguridad si la necesidad de este tipo de dispositivos existe o estaba ya cubierta. De dicho estudio se concluye que el dispositivo que se desea diseñar "*supone un desarrollo único y novedoso, siendo el primer dispositivo automático de ayuda para las tareas anteriormente descritas*" [1].

Esto llevará a la creación del concepto de PRESSMATIC, un proyecto cuyo objetivo es efectuar el diseño inicial de un dispositivo destinado a ayudar a personas cuya capacidad para realizar tareas manuales se ha visto reducida. El dispositivo ejecutaría de manera automática la función que desempeñan los dedos pulgar e índice en ciertas labores. Los resultados del diseño de dicho dispositivo dieron lugar a lo que se puede apreciar en la figura 1.

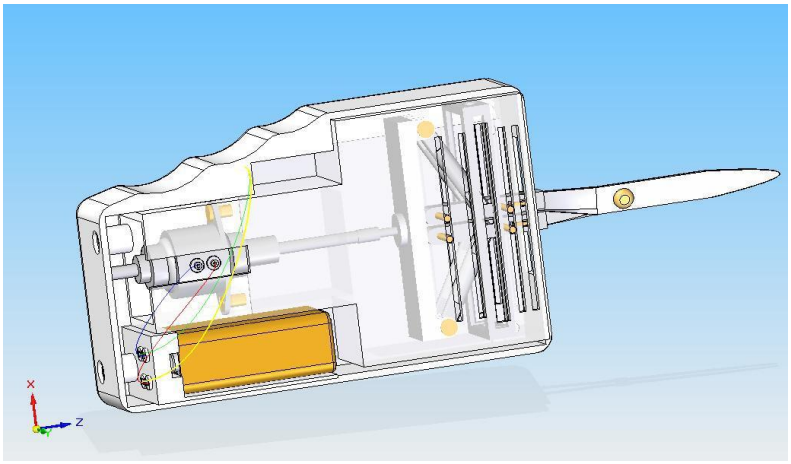

Figura 1. Dispositivo PRESSMATIC [1]

Cabe mencionar que el diseño inicial del dispositivo PRESSMATIC constaría inicialmente de un cuerpo principal y de un conjunto de cabezales intercambiables. En el cuerpo principal se encontrarían los principales subsistemas del dispositivo, de tal manera que pudiera ser sostenido con una mano, localizándose en éste una pantalla táctil que permitiera el control del mismo. Por otro lado, mediante los distintos cabezales se realizarían las distintas tareas que se deseen.

1.2. Introducción al proyecto

Posterior a la tesis de Gabriel surgen varios proyectos de fin de grado que tienen como finalidad crear un prototipo funcional y completo de PRESSMATIC [2]. En los distintos proyectos se desarrollarían las distintas partes o módulos del dispositivo como son la mecánica del mismo, que llevará a cabo los movimientos de los cabezales, así como el diseño de la electrónica de control del dispositivo. Cabe destacar que dos de estos proyectos de fin de grado, cuyos autores son Alonso Rosado [3] y Rodrigo Martín [4], progresarán de manera paralela y en coordinación con este proyecto, para integrar la comunicación con Pressmatic de una manera óptima.

Entre todos los posibles proyectos planteados se contemplaría la idea de realizar el control del dispositivo vía bluetooth mediante una aplicación existente en un dispositivo móvil. La finalidad del presente proyecto no es otra que llevar a cabo el desarrollo de esta aplicación contemplando los criterios de accesibilidad existentes, de tal forma que se pueda efectuar una utilización sencilla de la misma por cualquier usuario.

El empleo de la aplicación móvil se plantea como una posibilidad para controlar PRESSMATIC, aportando ventajas sobre la pantalla incorporada en el dispositivo. Entre las ventajas planteadas en un principio, destaca permitir una configuración más personalizada para el usuario con respecto a la pantalla táctil, ya que las plataformas actuales permiten un

gran margen tanto de personalización como de libertad a la hora de implementar una aplicación.

Por otro lado, la aplicación añadiría a su vez comodidad a la utilización del dispositivo, permitiendo al usuario manejar el dispositivo apoyando el Smartphone o Tablet en una superficie, sin la necesidad de interrumpir el trabajo actual para modificar los parámetros. El uso de la aplicación puede ser de gran utilidad también en situaciones en las que el acceso a la pantalla resulte más difícil para el usuario, como podría ser, por ejemplo, mientras se esté haciendo uso de una herramienta de PRESSMATIC.

Asimismo, mediante el desarrollo de la aplicación todos los usuarios de PRESSMATIC tendrán la posibilidad de sacar provecho de las funcionalidades que se proveen con su dispositivo móvil en comparación a la pantalla integrada. Cabe mencionar que, gracias a la instalación de la aplicación PRESSMATIC en este tipo de dispositivos, se facilitará en muchas ocasiones el control, ya que éstos suelen gozar de pantallas de mayor tamaño que la integrada en PRESSMATIC.

Estas y otras ventajas convierten así a la aplicación móvil en una firme alternativa al manejo de PRESSMATIC mediante su pantalla integrada, motivo por el cual se plantea y lleva a cabo este proyecto.

1.3. Objetivos del proyecto

Teniendo en cuenta que PRESSMATIC es un dispositivo innovador, todo lo relacionado con implementar un control sobre dicho dispositivo también lo será. Esto lleva a afirmar que la creación de una aplicación móvil para este tipo de dispositivos constituye una tarea distinta a lo que se haya podido ver anteriormente.

En el momento de llevar a cabo el diseño de la aplicación Android para PRESSMATIC se tendrá en cuenta principalmente la población a la que va dirigida y las necesidades que pueda tener este colectivo. En este caso, se tendrá que desarrollar una aplicación que sea **accesible** para todos y cuya utilización sea **simple** e **intuitiva**. Asimismo, la aplicación debería estar al alcance de todos, permitiendo su descarga por el mayor número de usuarios posible.

La aplicación permitirá a dichos usuarios **controlar** el dispositivo **PRESSMATIC** desde su dispositivo móvil mediante una **conexión** bluetooth que será fácilmente establecida por el usuario al iniciar la misma. Con el fin de que la aplicación sea eficaz y cumpla sus objetivos, se hace totalmente imprescindible que la conexión entre PRESSMATIC y el dispositivo Android sea **robusta** y **estable**.

Además, conocida la posible sofisticación de las aplicaciones Android que existen en la actualidad y, conocido el afán de integrar la accesibilidad en la aplicación, se propone también como posibilidad crear una **aplicación personalizable**. Ello implica que los

parámetros o características de la misma puedan ser modificadas por el usuario, como puede ser el idioma, la interfaz, el tamaño de fuente o de los botones, etc.

1.4. Fases de desarrollo

De manera esquemática, la metodología del proyecto se ha dividido en los siguientes bloques:

-Familiarización: En esta primera etapa se produce una toma de contacto con la tesis en la que se basa el proyecto. Se investigan temas como la accesibilidad en aplicaciones y se establecen los principales objetivos.

-Elección del sistema operativo: En esta parte se realiza un estudio del estado del arte de los sistemas operativos actuales. Se escogerá uno de ellos y se llevará a cabo la instalación del entorno de programación que se presente como el más adecuado.

-Programación: Este bloque es el que precisa una mayor cantidad de tiempo. En éste se lleva a cabo la comprensión y aprendizaje de conceptos relevantes al lenguaje de programación utilizado para implementar aplicaciones en el sistema operativo elegido. En esta etapa se crean los primeros proyectos y se efectúan las primeras pruebas de implementación en la plataforma escogida.

-Desarrollo e implementación: Cumplir el objetivo principal del proyecto mediante la creación de la aplicación es primordial. Con este fin, se pondrán en práctica todos los conceptos adquiridos en el bloque anterior y se hará necesario adquirir aquellos no cubiertos anteriormente a lo largo de un aprendizaje continuo.

-Documentación: Escribir este informe recopilando el proceso de desarrollo y las experiencias obtenidas para generar después las conclusiones y trabajos futuros que se expondrán más adelante.

Capítulo 2

Estado del arte

Este capítulo está destinado a ofrecer un análisis del estado de los avances tecnológicos actuales relacionados con el presente proyecto. Solo una vez conocidos estos datos será posible crear una aplicación a la altura de las expectativas, siendo capaz de cubrir las necesidades de los usuarios y del propio desarrollador.

2.1. Evolución tecnológica

2.1.1. Evolución de Smartphones y Tablets

En los últimos años se ha podido experimentar cómo ha ido evolucionando el mercado de los dispositivos móviles en concordancia con las necesidades de los usuarios. Con los avances actuales en la tecnología móvil se hace cada vez más común encontrar en cada hogar al menos uno de estos dispositivos por persona. Este crecimiento es debido a que este campo se encuentra en constante progreso, y es que las necesidades que son capaces de cubrir estos dispositivos aumentan día tras día.

En primer lugar, se hace necesario detallar lo que se entiende por dispositivo móvil. Un dispositivo móvil es un aparato de pequeño tamaño cuyas características principales son las siguientes [5]:

- Capacidades de procesamiento específicas.
- Memoria limitada.
- Conexión permanente o intermitente a una red.
- Diseñados para una función particular pero tienen capacidad para desarrollar otro tipo de funciones.
- Su posesión y manipulación se asocian a una persona, que puede configurarlos a su gusto.

El amplio campo de los dispositivos móviles abarca desde PDAs, reproductores mp3, navegadores GPS, Smartphones, hasta ordenadores portátiles y Tablets. Sin embargo, en este documento únicamente se dará cabida a los Smartphones y a las Tablets debido a que son éstos los que originan verdadero interés para el desarrollo del presente proyecto, por su relevancia dentro del campo de las aplicaciones móviles.

Figura 2.Dispositivos Smartphone y Tablet [6]

Con el crecimiento que se está produciendo en el campo de las aplicaciones para este tipo dispositivos móviles, denominadas comúnmente "Apps", se hace cada vez más frecuente el uso de dispositivos inteligentes. Esta expansión tiene su origen en el año 2007 cuando la compañía Apple provoca [7], con el lanzamiento de su Smartphone "iPhone", un nuevo planteamiento en el campo de las aplicaciones, permitiendo a desarrolladores externos y otras compañías correr sus aplicaciones utilizando su teléfono. Anteriormente los fabricantes no habían permitido a estos participar en el mercado, lo que llevó a que, a partir de ese momento, se iniciara el boom de las aplicaciones. A finales de 2008 se podían encontrar ya unas 500 aplicaciones en la App Store y 50 en Google Play (Android Market).

Actualmente se contabilizan más de 775.000 aplicaciones en la App Store y más de 800.000 en Google Play, principales mercados de aplicaciones del momento, seguidos del mercado de aplicaciones del Windows Phone, de Microsoft [8]. Estas aplicaciones permiten al usuario desenvolverse y llevar a cabo tanto tareas de la vida diaria como satisfacer necesidades de entretenimiento, ocio, comunicación, deporte... sacando provecho de las distintas funcionalidades del teléfono. Simplemente es cuestión de consultar nuestro dispositivo y comprobar la cantidad de aplicaciones que se encuentran a nuestra completa disposición para descargar y además, en la mayoría de ocasiones, de manera gratuita.

A consecuencia de esto y debido a las prestaciones que ofrecen los dispositivos móviles, el crecimiento de los mismos no ha hecho más que aumentar en los últimos años. Hoy en día, tenemos la posibilidad de encontrar una cantidad innumerable de modelos de teléfonos móviles y tabletas, que alcanzan cifras de usuarios cada vez mayores con el trascurso de los años, especialmente en el caso de los Smartphones (ver figura 3).

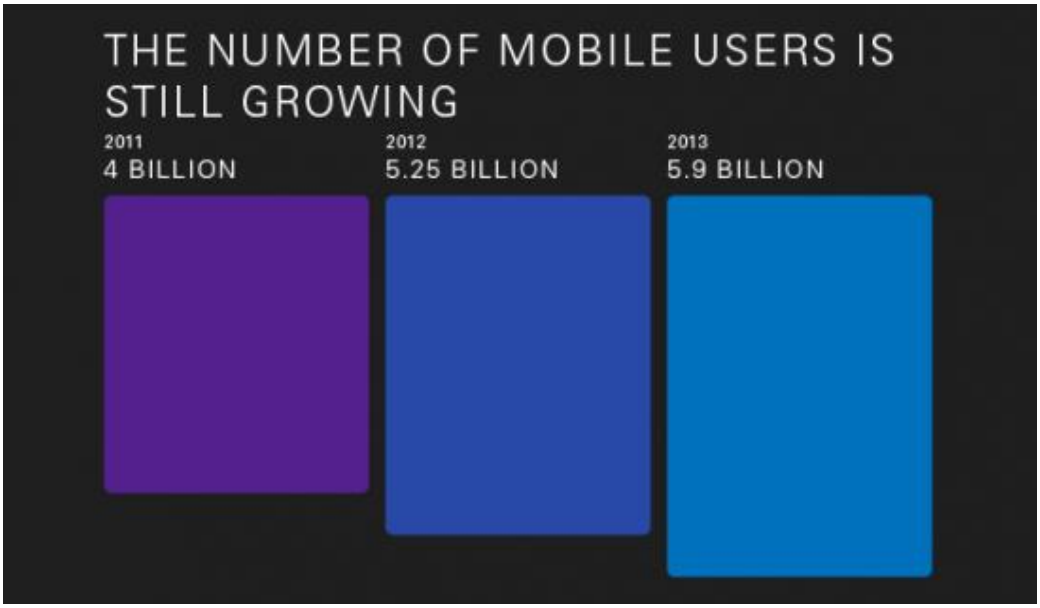

Figura 3. Crecimiento de los usuarios de telefonía móvil [9]

Es la influencia de este tipo de dispositivos en el mercado actual y el reciente desarrollo que ha habido en el campo de las aplicaciones lo que lleva a plantear la creación de una aplicación móvil que cubra la necesidad que nos ocupa, controlar un dispositivo que satisfaga necesidades de la vida diaria de una manera accesible. No obstante, para llevar a cabo el desarrollo de la misma, es necesario conocer el estado de la accesibilidad en aplicaciones a día de hoy.

2.1.2. Estado de la accesibilidad en aplicaciones

La accesibilidad se ha convertido en un concepto cada vez más presente y relevante en la vida diaria. Tal es su importancia que en la actualidad y desde hace unos años se hace cada vez más frecuente el deseo de diseñar cualquier objeto, servicio o lugar cuyo uso o utilización abarque el mayor número de usuarios posible, independientemente de las posibles limitaciones que estos puedan tener.

A partir de este interés surgen distintos conceptos de accesibilidad, como es el caso de la accesibilidad software que, tal y como se define en la Norma ISO 9241-171:2008 de Requisitos de accesibilidad del software: "se centra en los objetivos de maximizar el número de usuarios y aumentar el nivel de usabilidad experimentado por estos usuarios". Se puede deducir, por lo tanto, que el concepto de accesibilidad software está estrechamente relacionado con la usabilidad y será ésta la que determine en gran medida el grado de la misma.

En el campo de las aplicaciones móviles cabe mencionar que compañías como Apple mediante iOs y Google mediante Android principalmente, realizan esfuerzos por hacer de la accesibilidad un rasgo común de todas sus aplicaciones, facilitando en sus sistemas operativos el desarrollo de aplicaciones de este tipo.

Sin embargo, a pesar del deseo de conseguir que la accesibilidad forme gran parte de las vidas de las personas en el campo de las aplicaciones móviles, este objetivo queda todavía muy lejos de ser cumplido. El Grupo Fundosa estimó que, a mediados del 2013, el 99% de las aplicaciones que se pueden encontrar en el mercado no son accesibles en su totalidad, y que sólo una pequeña cantidad de desarrolladores ponen su empeño en hacer que esto cambie [10].

Otro de los estudios realizados en agosto del año 2013 por Discapnet, un portal especializado en personas con discapacidad, revela que las aplicaciones móviles más utilizadas como Facebook, WhatsApp, YouTube o Renfe **incumplen** los criterios de accesibilidad existentes [11]. Entre los criterios evaluados se pueden encontrar el acceso multidispositivo, cuyo fin es permitir a los usuarios utilizar cualquier dispositivo independientemente del sistema operativo, la accesibilidad global del sistema, que incluye utilizar elementos comunes y estándar para mejorar la compatibilidad y usar controles integrados en el sistema operativo, que están provistos del soporte de accesibilidad adecuado.

Si es verdad que, aunque la accesibilidad en aplicaciones no se haya implementado en la mayoría de aplicaciones, se pueden encontrar en el mercado una pequeña cantidad de ellas que sí contemplan los diversos criterios de accesibilidad existentes, denominadas "Apps" accesibles.

2.2. Aplicaciones accesibles

Entre las aplicaciones que satisfacen los criterios de accesibilidad podemos encontrar aplicaciones de todo tipo según las necesidades del usuario. Algunas de estas aplicaciones se detallan a continuación:

AudescMobile

Esta aplicación está desarrollada por la ONCE. Es de descarga gratuita y permite a los usuarios escuchar una descripción de las imágenes que se pueden observar durante la transmisión de una película o serie en la televisión o incluso en el cine de una manera sincronizada. Incluye también la posibilidad de detectar en qué parte se encuentra la película o serie en cuestión [12].

Figura 4. AudescMobile [13]

Figura 5. Voice Dream Reader [13]

Voice Dream Reader

Esta aplicación, en este caso de pago, permite al usuario la lectura de documentos de forma visual, de audio o de ambas al mismo tiempo. Contiene hasta 78 voces distintas en una veintena de idiomas [12].

TUR4all

Esta aplicación, desarrollada por PREDIF (Plataforma Representativa Estatal de Personas con Discapacidad Física), tiene almacenadas las condiciones de accesibilidad de más de 1800 establecimientos turísticos en todo el territorio español [14].

Figura 6. TUR4all

SIGNAME

SIGNAME es una aplicación que tiene como objetivo presentar de una forma sencilla y rápida vídeos, donde se muestran los distintos signos utilizados para la comunicación con los niños de centros de educación especial. Dichos vídeos han sido elaborados por expertos del Centro María Corredentora en Madrid [14].

Figura 7. SIGNAME

Como se puede observar, son varios los ámbitos en los que se pueden encontrar aplicaciones accesibles, en cambio, no ha sido posible encontrar ninguna aplicación accesible cuyo fin último sea el control de otro dispositivo, como es el caso de PRESSMATIC.

Por otro lado y, aunque no contemplen criterios de accesibilidad, sí se pueden encontrar aplicaciones que permiten controlar otros dispositivos desde un Smartphone o Tablet. Por ejemplo, existen varias aplicaciones destinadas a efectuar un control efectivo de la televisión como si el dispositivo móvil se tratara de un mando a distancia. Pero, sin embargo, serán aquellas que se valgan de una conexión bluetooth para llevar a cabo la comunicación las que realmente resulten de interés.

Tomando como referencia algunas de estas aplicaciones y, mediante la integración de distintos criterios de accesibilidad en la aplicación a desarrollar, ambos detallados más adelante, se convierte en una tarea necesaria la elección de la plataforma móvil a utilizar para comenzar el desarrollo.

2.3. Análisis del sistema operativo

A día de hoy se pueden encontrar una amplia gama de dispositivos móviles en el mundo. Cada vez es mayor el número de empresas que comercializan este tipo de productos debido al constante crecimiento de este mercado que, como se ha comentado anteriormente, se encuentra en continua expansión.

Entre las principales plataformas sobre las que se sustentan y trabajan este tipo de productos, se encuentran los sistemas operativos Android, iOs, BlackBerry y Windows Phone. Para tener una noción básica[1] de los sistemas operativos existentes se explican a continuación las principales diferencias de interés habidas entre ellos.

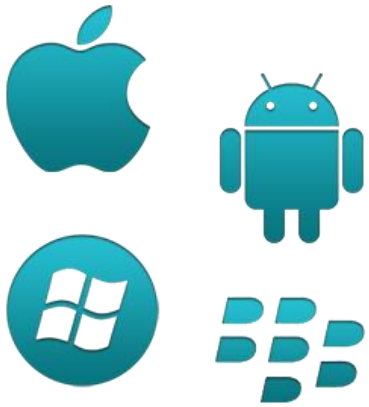

Figura 8. Principales S.O. para móviles [15]

- Android es un sistema operativo creado por Google y basado en Linux. Este sistema operativo está orientado principalmente a Smartphones y Tablets. Se trata de un sistema adaptable por los fabricantes y disponible en una extensa gama de dispositivos y precios [16].

  La programación de las aplicaciones Android se puede efectuar desde Mac, Linux y Windows y está escrita en lenguaje de programación Java, existiendo varios entornos de programación. Cabe destacar que cualquier desarrollador puede acceder al código fuente en el que se basan las aplicaciones, ya que Android se trata de una plataforma de código abierto y software libre. De esta manera, se consigue que los

[1] Cabe mencionar que no se detallarán, por ejemplo, datos acerca de qué plataforma resulta más rentable o qué tipo de usuarios tienden a pagar más por las aplicaciones. Esto es debido a que dichas características no se consideran de peso para la elección de la plataforma, entre otros motivos porque la aplicación se desarrolla sin ánimo de lucro.

componentes que se creen tengan como base componentes existentes en las librerías proporcionadas por Google, provocando que no exista una cantidad desorbitada de componentes, lo cual no tendría sentido.

Una de las grandes ventajas que podemos encontrar a la hora de desarrollar una aplicación Android es que existe una comunidad de desarrolladores muy competente, localizada principalmente en Stackoverflow [17]. Tal es así que a la hora de programar se puede encontrar solución de una manera sencilla a casi cualquiera de las dudas que le puedan surgir al desarrollador de una aplicación, dotando a éste de ejemplos de código.

Otra de las ventajas destacables de esta plataforma es que tiene a completa disposición de los usuarios un mercado de aplicaciones denominado Google Play, desde el cual se pueden descargar directamente las distintas aplicaciones de la comunidad al dispositivo móvil.

En cuanto a la accesibilidad en Android, existe una sección en la página de desarrolladores de Android dedicada exclusivamente a ella [18], facilitando pautas de accesibilidad para todos los usuarios pero centrándose más detalladamente en aquellas personas que sufren diversidad funcional visual.

- iOs es el sistema operativo creado, desarrollado y distribuido por la compañía Apple. Este sistema se desarrolló inicialmente para el iPhone para más tarde ser empleado por el resto de dispositivos de la compañía como iPad y iPod Touch.

  La programación en iOs está escrita en Objective-C y ésta se puede realizar únicamente mediante un ordenador de Apple, un Mac, lo cual no siempre está al alcance de todos. Por otro lado, una de las ventajas que ofrece trabajar con esta plataforma es el empleo del entorno de desarrollo proporcionado por Apple, X-Code. Este entorno tiene un consumo de memoria mucho menor que los principales entornos y, además, goza de un simulador mucho más eficiente que éstos [14].

  Si se trata el tema de la accesibilidad en iOs se puede afirmar que Apple, al igual que Android, provee a sus desarrolladores de la documentación necesaria para crear una aplicación accesible orientada también a la diversidad funcional visual. La integración de la misma en sus dispositivos es tan exitosa como en Android, liderando ambas plataformas el mercado en temas de accesibilidad.

- BlackBerry OS se trata del sistema operativo desarrollado por RIM (Research in Motion) únicamente para los dispositivos BlackBerry.

  La programación de estos dispositivos está basada en Java y se realiza desde un software proporcionado con la compañía, BlackBerry® Java® Development Environment [19].

Una de las principales características de esta plataforma es que está enfocada principalmente al ámbito empresarial, ya que permite gestionar de manera sencilla el correo electrónico, la agenda, tareas, notas… aunque los últimos dispositivos lanzados por la empresa se esfuerzan por englobar todas las necesidades de los usuarios.

En cuanto al mercado de aplicaciones de BlackBerry, denominado BlackBerry App World, se contabiliza que cuenta con unas 70.000 aplicaciones, número que quizás parezca pequeño en comparación a otras plataformas [20].

La accesibilidad en dispositivos BlackBerry se encuentra en desarrollo. Es apreciable que la compañía RIM está haciendo grandes esfuerzos por hacer que sus teléfonos integren funciones de accesibilidad en todos los sentidos, aportando soluciones para todo tipo de usuarios. Entre estas características se pueden observar rasgos parecidos a los que se podrían encontrar en otras plataformas y mejorando incluso algunas facetas en casos como la audición, la voz y la movilidad, sin hacer hincapié en ninguna en particular. Sin embargo, cabe destacar que muchos de estos atributos no se encuentran disponibles en la totalidad de dispositivos de la compañía [19].

Algo que diferencia a los dispositivos BlackBerry es que estos, sean antiguos o más modernos, incluyen a la hora de escribir el conocido teclado QWERTY de manera física o visible en la pantalla táctil. En el caso de los teclados físicos esto puede simplificar al usuario la tarea de escritura, dependiendo del caso.

- Windows Phone es el sistema operativo desarrollado por Microsoft. Es el descendiente directo de Windows Mobile pero, al contrario que éste, está orientado al mercado de consumo en vez de al mercado empresarial.

  La programación de estos dispositivos está escrita en C# y se efectúa mediante un software denominado Visual Studio.

  Entre las ventajas que presenta Windows Phone se puede encontrar una interfaz diseñada para ser especialmente intuitiva y una gran capacidad de sincronización con servicios de Microsoft [21]. Sin embargo, esta plataforma está peor valorada a la hora de tratar temas de personalización y realizar varias tareas al mismo tiempo.

  Microsoft ha desarrollado “Mobile Accesibility”, una aplicación que, según afirma la compañía, “permite a personas ciegas o que tienen problemas de visión utilizar Windows Phone 8 de una manera intuitiva y fácil” [22].

  Aun así, la empresa se encuentra actualmente detrás de Android e iOs a la hora de ofrecer soluciones de accesibilidad para las distintas necesidades de aquellos que más lo necesitan [23].

2.3.1. Criterios para la elección de la plataforma

Las características que debe poseer la plataforma que se va a utilizar para llevar a cabo el desarrollo de PRESSMATIC son concretas y su elección dependerá de las mismas. Dichas características se han determinado en base a las necesidades actuales y futuras de la aplicación y, por lo tanto, de desarrolladores y usuarios. Éstas son las siguientes:

- Característica 1: Capacidad para adaptar los criterios de accesibilidad que sean necesarios en la aplicación de una manera simple, facilitando la creación de una aplicación cuyo uso resulte intuitivo para el usuario.

- Característica 2: Facilidad para los desarrolladores para llevar a cabo el desarrollo de la aplicación mediante el empleo de las herramientas y entornos de programación existentes.

- Característica 3: Disponibilidad sencilla para todos los usuarios que vayan a usar el dispositivo PRESSMATIC y capacidad de utilización por una gran variedad de dispositivos, posibilitando así su uso a la mayor cantidad de usuarios posible.

- Característica 4: Posibilidad de realizar pruebas de funcionamiento de la aplicación que, en el caso de PRESSMATIC, requerirá el empleo de la función bluetooth en dispositivos reales.

- Característica 5: Una evolución constante de la plataforma favorecerá la inclusión de mejoras en la aplicación y el desarrollo de nuevas ideas.

Teniendo en cuenta los anteriores criterios y conociendo brevemente las distintas plataformas se hace posible obtener una serie de conclusiones. Éstas se tomarán como pautas para efectuar la selección de la plataforma más adecuada que permita llevar a cabo el desarrollo de la aplicación PRESSMATIC. Dichas conclusiones son las siguientes:

- Conclusión 1: Como se ha comentado anteriormente, son iOs y Android las plataformas más avanzadas en términos de accesibilidad, proporcionando directrices a los desarrolladores para hacer de la accesibilidad un rasgo habitual en sus aplicaciones.

- Conclusión 2: A la hora de desarrollar una aplicación para cualquier plataforma es sencillo encontrar tutoriales tanto de programación como de instalación de los diversos entornos de programación, siendo iOs y Android los más extendidos en este aspecto. Por otra parte, es iOs la única plataforma que requiere de un tipo específico de ordenadores para realizar el desarrollo de aplicaciones y, en general, se trata de ordenadores de mayor coste económico.

- Conclusión 3: Si se tiene en cuenta la cantidad de usuarios que utilizan hoy en día las distintas plataformas, se puede determinar que Android e iOs son las plataformas

con más éxito. En cambio, es la plataforma Android la que lidera el mercado con una cuota de cerca del 79% en el año 2013, como se puede observar en la tabla 1, y será ésta, por lo tanto, con la que se llegará a un mayor número de usuarios. Además, Android, como se ha mencionado anteriormente, dispone de una cantidad muchísimo más amplia de dispositivos entre los que elegir, favoreciendo así las limitaciones de los usuarios y sus necesidades.

Global Smartphone Operating System Marketshare %	Q4 '12	2012	Q4 '13	2013
Android	70.3%	68.8%	78.4%	78.9%
Apple iOS	22.0%	19.4%	17.6%	15.5%
Microsoft	2.7%	2.7%	3.2%	3.6%
Others	5.0%	9.1%	0.7%	2.0%
Total	100.0%	100.0%	100.0%	100.0%

Tabla 1. Cuota de mercado de los Smartphone según el SO [24]

- Conclusión 4: Es posible llevar a cabo pruebas de simulación en cualquiera de los entornos de programación de las distintas plataformas. A pesar de ello, cuando se realizan test de conectividad bluetooth se hace esencial poseer un dispositivo físico, ya que se hace imposible la simulación de dicha conexión. Es por este motivo por el que se debe disponer de un dispositivo de la plataforma seleccionada.

- Conclusión 5: Como ya se ha visto, las plataformas se encuentran en constante evolución, adaptándose continuamente a las necesidades de los usuarios, con el fin de mantenerse en el mercado de una manera competitiva. Conocido esto, se podría afirmar que son Android, iOs y Windows Phone las que se desarrollan con mayor efectividad en concordancia con las nuevas necesidades que surgen en los usuarios.

2.3.2. Plataforma seleccionada

De las conclusiones previas podemos deducir que las plataformas que mejor se adecuarán a las necesidades surgidas del desarrollo de la nueva aplicación que controlará el dispositivo PRESSMATIC mediante bluetooth son iOs y Android. Estas plataformas se encuentran a la cabeza en temas de accesibilidad móvil, están ampliamente extendidas y la evolución que han experimentado en los últimos años es creciente.

Otro de los factores importantes que se ha tener en cuenta es que la aplicación debe de ser adquirible por cualquier usuario mediante los mercados de aplicaciones existentes. Esto es posible en cualquiera de las plataformas, no obstante, es la plataforma Android la que posee una mayor cuota de mercado, aumentando así la probabilidad de que el usuario sea poseedor de un terminal Android y no necesite adquirir otro.

Por otro lado el sistema operativo Android se encuentra, en comparación a iOs, en una mayor cantidad de terminales de distintas compañías a elección del consumidor. Por ello, se dispondría mediante su elección, de una mayor diversidad de precios en el caso de no poseer un Smartphone o Tablet, y tener la necesidad de adquirir uno.

Estos y otros motivos, como el grado de personalización que ofrece la plataforma por encima de otras y la gran comunidad de desarrolladores que posee, convierte a **Android** en la **plataforma preferida** para llevar a cabo el desarrollo de la aplicación móvil , denominada PRESSMATIC, cuyos objetivos se han detallado en el apdo. 1.3.

Figura 9. Logo plataforma Android [25]

A pesar de todo, aunque Android haya sido la plataforma seleccionada para desarrollar una aplicación prototipo por los criterios anteriormente descritos, es primordial tener en cuenta que uno de los objetivos del dispositivo PRESSMATIC es llegar al mayor número de usuarios posible. Esto incluye tanto el hecho de abarcar la máxima cantidad de usuarios como incluir en la aplicación el concepto de "accesibilidad para todos". Sabido esto, no se debe descartar el desarrollo de la aplicación PRESSMATIC para el resto de plataformas.

2.4. Comparativa y elección del entorno de programación

Con el propósito de crear la aplicación, la primera acción que se debe emprender es la instalación del entorno de desarrollo. A día de hoy existen varias posibilidades a elección de cualquier desarrollador, de tal manera que se pueda escoger entorno que más se adecúe a las necesidades de los mismos o del proyecto.

En el caso de la aplicación PRESSMATIC se ha procedido a utilizar "IntelliJ IDEA", de la empresa JetBrains. Su elección ha sido debida a que se trata de un entorno de desarrollo que plantea varias ventajas con respecto a otros entornos conocidos, como pueden ser Eclipse o Netbeans, sus competidores más directos.

Figura 10. Logo IntelliJ IDEA [26]

El motivo principal para elegir IntelliJ IDEA es la mejora de productividad que ofrece [27], permitiendo a los desarrolladores usar una amplia variedad de herramientas y facilidades que agilizan la programación y diseño. Entre éstas se pueden encontrar comprobación y análisis de código según se escribe, funciones de asistencia y accesibilidad para la generación de código, gran diversidad de refactorizaciones, que permiten optimizar código aplicando cambios en la estructura del mismo, facilitando así un mantenimiento posterior, un compilador más ágil, personalización de la interfaz de trabajo, etc.

Además de lo mencionado, IntelliJ IDEA dispone de un generador de interfaz en tiempo real que facilita el trabajo, sobre todo en los casos en los que la interfaz tiene una mayor importancia. Esta particular de este entorno posibilita la visualización de los elementos de la interfaz con los que se trabaja en cada momento, enmarcando dichos elementos con un recuadro azul. Esta característica ha sido de ayuda para el proyecto y se puede observar en la figura 11.

Figura 11. Visualización de los elementos

A pesar de que IntelliJ IDEA está diseñado para programar proyectos a muy gran escala y aprovechar así las mejoras de productividad previamente descritas, las facilidades que éste ofrece para el nivel de programación que es requerido en el proyecto actual son perfectamente aplicables, siendo por tanto la mejor opción a escoger.

A partir de ahora y a lo largo del desarrollo del presente proyecto, se utilizará esté entorno en su versión "**IntelliJ IDEA Community Edition 13.0.2**". Se puede encontrar una guía de instalación de este entorno en el Anexo A.

Las pruebas y la visualización de las distintas pantallas de la interfaz de usuario se podrán observar a la vez que se lleva a cabo el proceso de diseño pero esto no es suficiente, ya que el software, aunque nos permite hacer simulaciones en distintos dispositivos, aporta una reproducción que no es totalmente precisa. Por este motivo y debido a que es imprescindible poseer un terminal para realizar los ensayos del protocolo bluetooth, se empleará un móvil "**Samsung Galaxy S Advance GT-i9070**" cuya **versión** de **Android** es **2.3.3 (nivel API 10**) para efectuar las pruebas, mostrado en la figura 12.

Figura 12. Samsung Galaxy S Advance [6]

Cabe mencionar que la versión de Android más moderna es la versión 4.4 (nivel API 19) y que la versión utilizada data de hace más de un par de años. Esto se ha considerado un punto a favor, ya que la correcta implementación de la aplicación en móviles que disponen de menores prestaciones, como puede ser una pantalla de menor tamaño o un procesador más lento, llevará a un adecuado funcionamiento de la misma en versiones posteriores.

De todos modos, la aplicación final será testeada en la medida de lo posible en versiones más modernas con otros dispositivos.

Capítulo 3

Plataforma Android

Este apartado está enfocado a explicar al lector de una manera genérica las características que definen a la plataforma Android como sistema operativo, con el objetivo de tener una visión básica del mismo que va desde la arquitectura del sistema hasta la estructura de una aplicación.

3.1. Arquitectura del sistema

La arquitectura Android está basada en el kernel de Linux de tal manera que incorpora programas de gestión de memoria y ajustes de seguridad, entre otros servicios, y está formada por cuatro capas basadas todas ellas en software libre. Cada una de estas capas utiliza servicios de las demás permitiendo a la vez al resto de capas utilizar sus servicios [28].

A pesar de que Android esté basado en Linux, el usuario nunca interaccionará directamente con Linux ni tampoco lo harán las aplicaciones instaladas en el dispositivo, sino que será el propio sistema operativo el que lo haga. En la figura 13 se presenta la estructura que sigue un sistema Android y, justo después, se describen las capas que se pueden encontrar en el diagrama, en orden descendente.

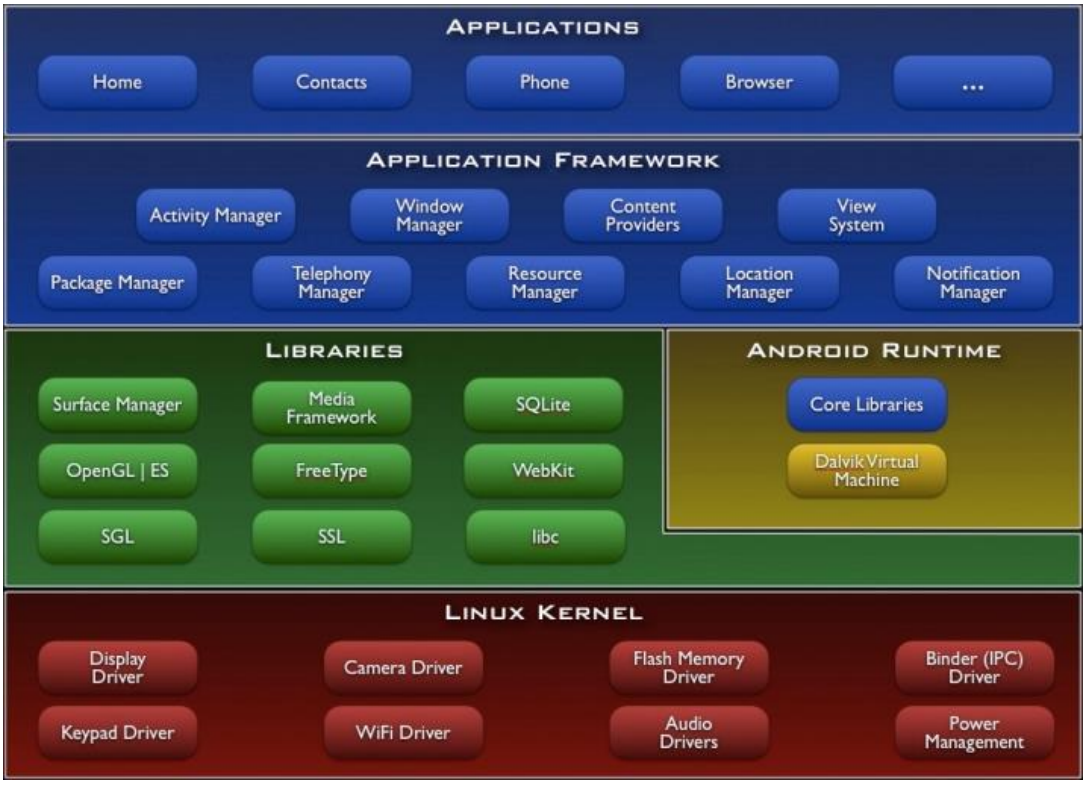

Figura 13. Arquitectura de Android [29]

- Aplicaciones: es el nivel superior y está formado por el conjunto de aplicaciones instaladas en un dispositivo Android, hayan sido instaladas por el usuario o preinstaladas por el sistema. Todas las aplicaciones harán uso de servicios, API y librerías presentes en otros niveles.

- Entorno de aplicación: representa las herramientas de desarrollo libre para crear aplicaciones innovadoras y variadas, los bloques de construcción de alto nivel. Esta capa ha sido diseñada de tal manera que se permita a cualquier desarrollador, compañía o usuario, la reutilización de componentes de una manera mucho más sencilla, mediante la utilización de las mismas API [28].

  Entre las API más importantes se pueden encontrar:

  -Vistas: provee elementos (vistas) para construir las interfaces de usuario.
  -Gestor de ventanas: encargado de administrar las ventanas de las aplicaciones.
  -Gestor de recursos: gestiona los recursos de la aplicación que no tienen relación con el código en sí.
  -Gestor de actividades: encargado de controlar el ciclo de vida de las aplicaciones en Android.
  -Gestor de notificaciones: gestiona las alertas que se muestran en la barra de estado.

-Proveedores de contenido: proporciona datos existentes en otras aplicaciones como, por ejemplo, un número de teléfono o un mensaje.

- Librerías: esta capa incluye un conjunto de librerías en C/C++ utilizadas por Android. Están compiladas en el código nativo del procesador y constituyen, junto con la capa inferior, la base de Android. Algunas de estas librerías son[30]:

    - Webkit: proporciona un navegador web empleado en el navegador de Android. Es la misma librería que usan navegadores de otras compañías como Google o Apple.
    - Códecs multimedia: esta librería contiene los códecs necesarios para reproducir y grabar multitud de formatos de audio, imágenes y video como AAC, AVC, MP3, MP4, JPG y PNG.
    - Surface Manager: gestiona los distintos elementos de representación y navegación en pantalla.
    - Gestor de gráficos 2D y 3D: sustentan la capacidad gráfica de Android. Las librerías 3D están basadas en OpenGL permitiendo utilizar el acelerador 3D si está disponible o el software altamente optimizado de proyección 3D.Por otro lado SGL provee gráficos 2D y es por ello la más utilizada.
    - FreeType: proporciona fuentes en bitmap y renderizado vectorial.
    - SQLite: potente gestor de bases de datos relacionales para aplicaciones.
    - SSL: posibilita la utilización de los servicios de encriptación Secure Socket Layer para establecer comunicaciones seguras.

- Runtime de Android: se trata del entorno de ejecución de Android. Esta capa está provista de la máquina virtual Dalvik. Esta máquina virtual es una máquina de java optimizada especialmente para dispositivos con recursos limitados.

- Núcleo Linux: se trata del núcleo de Android, su nivel inferior, integrado por el SO Linux en su versión 2.6. Este núcleo suministra servicios como gestión de memoria, pila de protocolos, seguridad y soporte de drivers necesarios para que los distintos dispositivos puedan ser utilizados correctamente.

3.2. Ciclo de vida de una Actividad

Android es distinto a otros S.O. en términos del ciclo de vida de una aplicación. La mayor diferencia estriba en que el ciclo de vida de éstas es controlado por el sistema en lugar de por el usuario.

Una aplicación está compuesta por un conjunto de elementos básicos de visualización denominados **actividades**. Cada una de estas actividades tiene un ciclo de vida dentro de la aplicación y son éstas las que controlan el funcionamiento de las aplicaciones. Al navegar por una aplicación se realizan cambios de actividades y, al efectuar dichos cambios, el

sistema almacena información acerca de las actividades visualizadas con anterioridad, de tal manera que el usuario pueda regresar a las mismas utilizando la tecla "atrás".

Las aplicaciones Android van asociadas a un proceso Linux que las contiene y dicho proceso seguirá existiendo hasta que el sistema precise la memoria asignada a esta aplicación [28]. En relación a esto, un rasgo importante de Android es que será el sistema el encargado de destruir un proceso dependiendo de la importancia para el usuario y de la capacidad de memoria disponible en un momento concreto. Si después de destruir el proceso el usuario ejecuta la aplicación de nuevo, el proceso será creado otra vez, volviendo al estado inicial de la misma.

Sabido que son las actividades las que determinan de una manera u otra el estado de las aplicaciones, a continuación se describen los distintos estados por los que se compone el ciclo de actividad de las mismas:

-Activa (Running): La actividad es visible y tiene el foco[2].
-Visible (Paused): La actividad deja de tener el foco debido a que hay otra actividad que pasa a activa.
-Parada (Stopped): Este estado se da cuando la actividad deja de ser visible. En este estado es recomendable almacenar el estado de la actividad.
-Destruida (Destroyed): La actividad se finaliza al invocarse el método `finish()` o debido a que el sistema Android la elimina.

La comprensión de estos estados es básica para el desarrollo de una aplicación, ya que sólo así será posible implementar correctamente los métodos que capturan y manejan los cambios de estado.

La figura 14 muestra, de manera escalonada, el ciclo de vida de la actividad, presentando para cada uno de los estados, la llamada al correspondiente método del ciclo de vida que llevará la actividad un escalón arriba, hasta hacerla visible para el usuario, o abajo, hasta cerrarla:

[2] Si una actividad tiene el foco quiere decir que el usuario puede interactuar con ella de manera directa. En cambio, si una actividad ha perdido el foco la actividad ésta puede ser todavía visible, pero esto significará que una nueva actividad, ya sea transparente o que no ocupa toda la pantalla, ha aparecido.

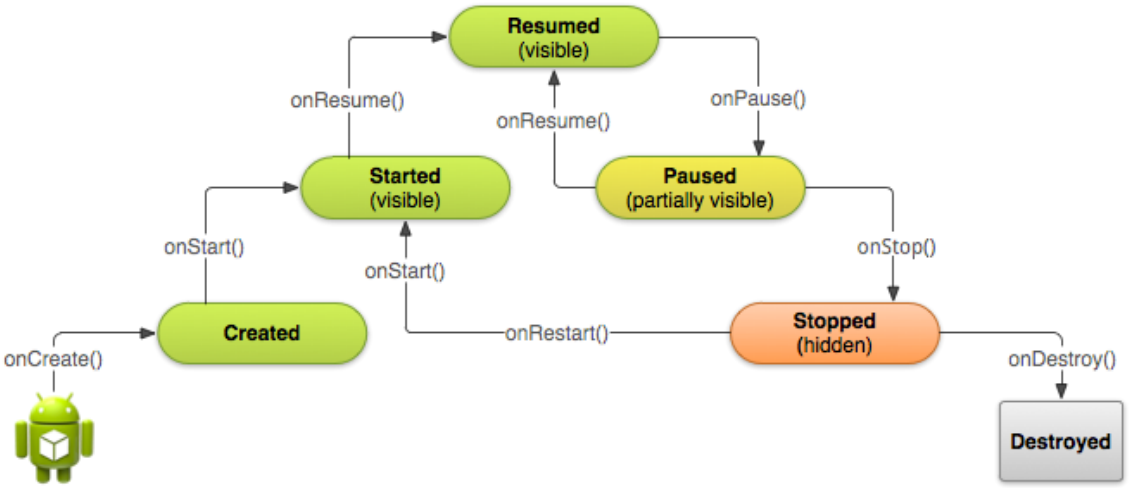

Figura 14. Métodos llamados en el ciclo de vida y estados asociados [18]

Los métodos llamados a lo largo del ciclo de vida de la actividad son [17][28]:

-onCreate (Bundle): Se llama cuando se crea la actividad. Aquí es donde se realizan las inicializaciones tales como la creación de la interfaz de usuario o asociación de datos a las estructuras o listas correspondientes.
-onStart(): Se llama a este método antes de que la actividad sea visible para el usuario.
-onRestart(): Este método es llamado después de que la actividad haya pasado por el método onStop.
-onResume(): Llamado cuando la actividad comienza la interacción con el usuario. En este punto la actividad está activa.
-onPause(): Se llama cuando el sistema está a punto de lanzar una nueva actividad y, por ello, la actividad actual pasará a segundo plano. Este método es típicamente usado para convertir cambios no guardados en datos persistentes, parar animaciones y detener procesos que puedan consumir memoria.
-onStop(): Este método es llamado cuando la actividad deja de ser visible para el usuario debido a que otra actividad ha ocupado su lugar.
-onDestroy(): Este es el último método que se llama antes de que la actividad sea destruida. Esto puede deberse a varias causas: si el usuario pulsa el botón "Volver", si se produce una llamada al método `finish()` o en el caso de que el sistema precise de memoria.

Como se puede ver, son varios los métodos empleados para gestionar estos estados y es necesario aclarar que no hace falta implementar todos y cada uno de ellos en las distintas actividades. En particular, el único realmente imprescindible en todo ciclo de vida es el método *onCreate()*. Si alguno de los métodos no se implementa se hará la llamada a dicho método, pero el sistema no llevará a cabo ninguna acción específica. Sin embargo, en algunas ocasiones es altamente recomendable hacer uso de dichos métodos, principalmente para tener un mayor control sobre las actividades y lograr que la aplicación se comporte tal y como se desea.

3.3. Elementos de la interfaz de usuario. Vistas.

La interfaz de usuario de Android consiste en una jerarquía de objetos descendientes de la clase *View* (vista), definidos como una subclase. Cuando se trata de organizar los distintos componentes que forman parte del diseño de una pantalla, se dispone de distintos elementos combinables, denominados *Layout*, que permiten contener otros elementos de tipo *View* y realizar la distribución de los mismos de la manera deseada. Entre estos *Layout* los más utilizados en la plataforma Android son los siguientes [18]:

-Linear Layout: organiza sus elementos (hijos) en una única fila horizontal o vertical, facilitando una barra de desplazamiento si éstos exceden la longitud de la pantalla.
-Relative Layout: especifica la localización de los elementos adoptando estos una posición relativa unos con respecto a otros.
-Table Layout: dispone los elementos de forma tabular.
-Absolute Layout: posiciona los distintos elementos de forma absoluta, especificando su posición exacta en la pantalla mediante coordenadas.
-Frame Layout: permite que los elementos que contiene cambien su visibilidad de manera dinámica.

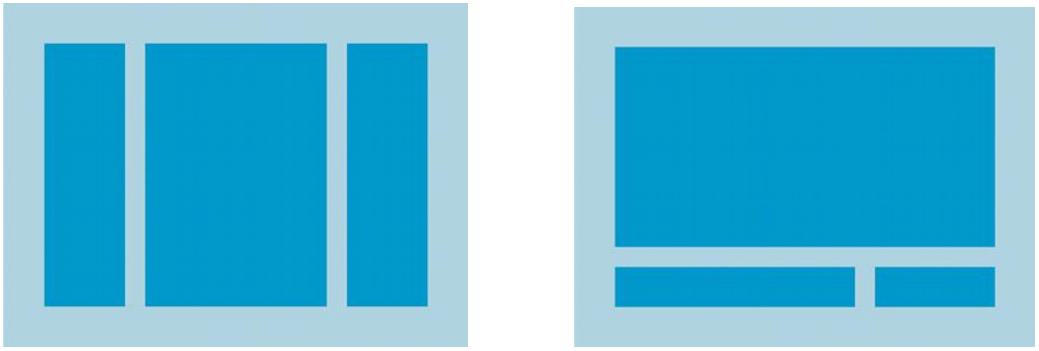

Figura 15. Ejemplos de Linear Layout y Relative Layout

Los distintos componentes de tipo *View* que pueden formar parte de la interfaz de usuario, mediante los cuales los usuarios obtienen información y se comunican con la aplicación, pueden ser de tipos diversos. Los elementos que se suelen encontrar con más frecuencia en una aplicación son los botones de distintos tipos (de tipo *Button*), y los textos (de tipo *TextView*), ya sean editables o no, junto a muchos otros como barras de progreso, menús desplegables, listas, etc.

Junto con cualquier componente derivado de la clase *View* hay asociados una serie de **atributos** que definirán las propiedades del mismo. Existen una serie de propiedades que se deben definir obligatoriamente para cada componente, como pueden ser el ancho (`layout_width`) y alto (`layout_height`) de la vista, y propiedades opcionales, como márgenes, color o etiquetas que se quieran poner en la vista. Según el atributo que se esté definiendo los valores o estados que se establecen serán diversos, por ejemplo, al parámetro

de altura le corresponderá un valor determinado y a un parámetro de texto una serie de caracteres.

Debido a la importancia que tiene la interfaz en el proyecto actual, algunos de los atributos más utilizados para definir los distintos elementos de la interfaz de usuario de PRESSMATIC se clarifican a continuación [18]:

-`id`: asocia un recurso ID con el elemento, una identificación, pudiendo utilizarlo así para referirse al mismo a la hora de disponer los elementos en la pantalla o de programar sus funciones.
-`orientation`: en un layout, especifica si la disposición de los elementos de la vista será vertical u horizontal.
-`layout_width`: determina la anchura de la vista, pudiendo establecer una cantidad o unos determinados parámetros que variarán según la pantalla del dispositivo. Estos parámetros son: "match_parent", que produce que la vista ocupe todo el espacio disponible para ese atributo, por ejemplo, toda la anchura, o "wrap_content", que significa que la vista se ajustará al contenido. Es un atributo requerido, ha de definirse siempre.
-`layout_height`: establece la altura de la vista, pudiendo adoptar un valor cuantitativo o uno de los parámetros descritos, como para la anchura. Es un atributo requerido, ha de definirse siempre.
-`layout_margin***`: según el caso (Right, Left, Top, Bottom), establece la distancia del elemento de la vista a uno de los márgenes de la pantalla o a otra vista.
-`layout_to****Of`: en un *RelativeLayout*, posiciona la vista en cuestión a un lado u otro (Right, Left) de una vista cuyo atributo `id` es conocido.
-`layout_above` o `layout_below`: tienen la misma función que lo anterior, pero situando en este caso la vista en cuestión encima o debajo de la otra vista.
-`background`: determina el fondo de la vista, pudiendo tomar como valor un color, definido o establecido mediante un código de colores (ver Apdo 4.2.6), o una imagen.
-`layout_weight`: cuantifica la importancia que se le da a la vista asignándole un valor y, empleando ese valor, se realizará una asignación de la cantidad de espacio que ocupa dicha vista.
-`weightSum`: indica el máximo valor que pueden sumar los atributos `weight` de un layout. Si por ejemplo definimos un `weightSum` de 1.0, y asignamos a una vista hija un valor de weight de 0.5, ésta ocupará la mitad del espacio disponible. Su valor debe ser decimal. Si este valor no se especifica su valor será la suma de los `weight` de las vistas hija.
-`text`: establece el texto que aparecerá en la vista.
-`textSize`: define el tamaño de fuente de la vista.
-`textColor`: determina el color de la fuente utilizada.
-`textStyle`: realiza la elección del estilo de la fuente usada.
-`gravity`: define cómo se posiciona la vista con respecto al layout que la contiene. Este atributo puede tomar valores como top, bottom, left, right, center_vertical, center_horizontal, etc.

`-contentDescription`: este atributo hace uso de uno de los servicios de accesibilidad prestados por Android emitiendo una descripción sonora del elemento en cuestión, definida en este atributo, cuando la función Talkback está activada.

Conocidos estos atributos, la interpretación del código de los archivos que implementan la interfaz se convierte en una tarea más sencilla.

3.4. Principales componentes de una aplicación Android

Mediante la SDK de Android la compañía Google proporciona a los desarrolladores de una manera sencilla y gratuita todo lo necesario para el desarrollo de aplicaciones en esta plataforma.

La SDK de Android pone, a disposición de todos, una serie de elementos básicos que resultan de vital importancia para desarrollar aplicaciones. Los componentes más importantes se describen a continuación con el objetivo de ofrecer al lector una idea básica, aunque algunos se han descrito ya o se describirán más adelante de una manera más detallada si se considera necesario [28]:

- Actividad (Activity): como ya se han descrito anteriormente, una aplicación estará formada por una serie de pantallas que contienen distintos elementos de visualización. Cada una de estas pantallas es lo que se conoce como una actividad en Android[3] y su finalidad principal es la creación de la interfaz de usuario. Cada una de estas actividades heredará sus funcionalidades de la clase Activity.

- Vista (View): esta clase representa el bloque principal para construir los distintos elementos que componen las interfaces de usuario de una aplicación, como pueden ser un botón o un cuadro texto. Todos los elementos serán descendientes de la clase View y podrán ser definidos mediante XML o utilizando código Java. Si se utiliza XML, se crea la definición de una pantalla de manera parecida a como se haría con una página web.

- Layout: como se ha visto antes, un elemento de este tipo permite agrupar un conjunto de vistas de una determinada forma según su tipo. Los objetos de este tipo son descendientes de la clase View y también se pueden definir en código Java o utilizando XML.

- Servicio (Service): es un componente de las aplicaciones que se ejecuta sin que el usuario interactúe con él, en segundo plano. Se utiliza para desempeñar una operación de mayor duración o simplemente para aportar funcionalidades que puedan utilizar otras aplicaciones. Se ejecuta, en general, a la vez que el proceso principal de la aplicación en la que se contiene.

[3] A partir de este momento se hablará indistintamente de pantallas y de actividades puesto que representan el mismo concepto.

- Intención (Intent): es una descripción de una operación que se desea realizar, la voluntad de querer llevarla a cabo. Entre las intenciones destacan cuatro tipos de operaciones, según la operación que efectúan:

  -Lanzar una actividad.
  -Lanzar un servicio.
  -Lanzar un anuncio tipo broadcast.
  -Comunicarse con un servicio.

3.5. Estructura de un proyecto Android

Como se ha comentado anteriormente, se ha escogido como entorno de desarrollo "IntelliJ IDEA". En esta sección se detallará la estructura de un proyecto Android creado con esta herramienta de desarrollo, escogida por motivos anteriormente descritos.

Para desarrollar un proyecto la primera acción a seguir es llevar a cabo la creación de dicho proyecto. El proceso de creación depende de las características que se quieran asignar a un proyecto. Se puede encontrar una guía de creación de proyectos en el Anexo B. Después de crear un proyecto, en el ejemplo denominado "Hola Mundo" se generarán los siguientes archivos, estructurados por carpetas:

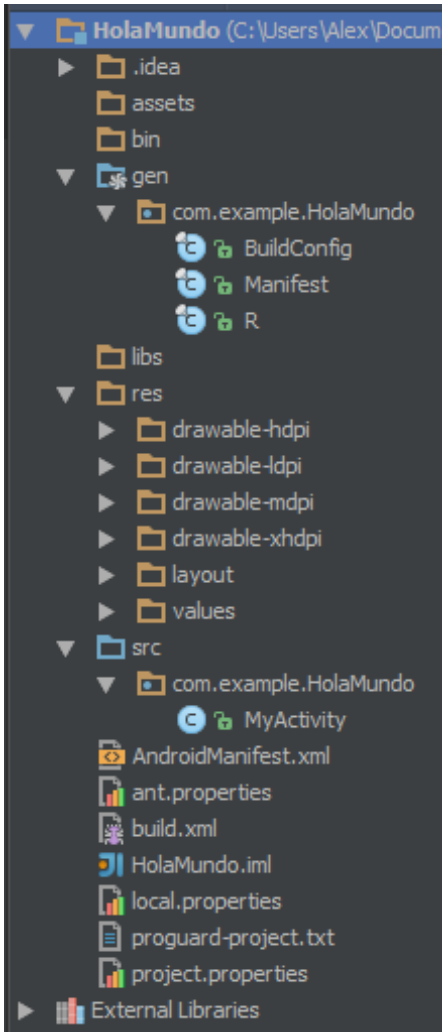

Figura 16. Archivos existentes al crear un proyecto

A grandes rasgos las funciones que cumplen los principales archivos que se pueden encontrar son las siguientes [31]:

-*AndroidManifest.xml*: es un descriptor de la aplicación, mediante este archivo se determinará la configuración general de la aplicación como los permisos que necesita, las actividades existentes, intenciones, la versión mínima de Android necesaria para ejecutarla, etc. Este archivo es de suma importancia en un proyecto Android y el que se encuentra en este caso en la aplicación PRESSMATIC se adjunta en la hoja siguiente.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example.AplicacionProyecto"
          android:versionCode="2"
          android:versionName="2.0">
<!--Determina la mínima versión de Android para utilizar la
aplicación. En este caso coincide con la versión que usa el terminal
de pruebas -->
    <uses-sdk android:minSdkVersion="10"/>

<!--  Determina los permisos necesarios
para la ejecución de la aplicación -->
      <uses-permission android:name="android.permission.BLUETOOTH"/>
      <uses-permission
      android:name="android.permission.BLUETOOTH_ADMIN"/>

<!--  Declaración de la aplicación (con su nombre) y de las
actividades y    servicios que contiene -->

    <application  android:label="@String/app_name"
                  android:icon="@drawable/appicon7"
                  android:name=".MyApplication">

        <service android:name=".BluetoothConexion"/>

        <activity android:name=".PantallaInicioSplash"
                  android:label="@String/splash">
            <!-- Determina que la actividad inicial
            de la aplicación será ésta -->
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category
                  android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>

        <!-- El parámetro android:label hace uso de la
        herramienta Talkback de accesibilidad de Android
        para emitir un mensaje cuyo contenido es dicho label -->
        <activity android:name=".Pressmatic"
                  android:label="@String/LabelMainActivity"/>
        <activity android:name=".AcercaDe"
                  android:label="@String/LabelAcercaDe"/>
        <activity android:name=".ModoContinuo"
                  android:label="@String/LabelModoContinuo"/>
        <activity android:name=".ModoPasoPaso"
                  android:label="@String/LabelModoPasoPaso"/>
        <activity android:name=".ModoAgarreInst"
                  android:label="@String/LabelModoAgarreInst"/>
        <!-- La lista de dispositivos aparecerá como un cuadro de
        diálogo (Dialog) -->
        <activity android:name=".ListaDispositivos"
                  android:label="Lista de dispositivos"
                  android:theme="@android:style/Theme.Dialog"/>
        <activity android:name=".Preferencias"
                  android:label="Preferencias"/>
    </application>
</manifest>
```

-*ant/project/local.properties*: los archivos de este tipo serán utilizados por IntelliJ IDEA y no deben ser editados. Se utilizan para comprobar la versión del API y otras características.
-*proguard.cfg*: en este archivo se edita la configuración de optimización de código y la ofuscación cuando la aplicación sea publicada

Por otro lado, las funciones que cumplen las principales carpetas son las siguientes:

-*gen*: es una carpeta que no se debe modificar, se genera automáticamente por el sistema y en ella se encuentran el paquete creado y el archivo R.java. Este archivo se encarga del manejo interno de recursos, asociándolos con sus identificadores correspondientes de tal manera que puedan ser accedidos desde java.
-*assets*: en ésta se hallan los archivos de recursos requeridos y utilizados por la aplicación. Esta carpeta nunca se modifica, al contrario que la carpeta res.
-*src*: dentro de esta carpeta se ubica el paquete creado y el código fuente de la aplicación en su conjunto. En ella aparecerá la actividad "Hello World!" con el nombre que se le haya dado si se escoge dicha opción en la creación del proyecto.
-*res*: los recursos de la aplicación son modificados en esta carpeta mediante el empleo de varias subcarpetas:

- *drawable*: existen cuatro carpetas de este tipo donde se encuentran los distintos gráficos de la aplicación como las imágenes que contienen los botones e iconos. Según el tamaño de los mismos serán incluidos en una carpeta *drawable* u otra.
- *layout*: en esta carpeta se localizan archivos de las vistas de la aplicación en formato XML. Estos determinarán la apariencia y presentación de las distintas vistas que aparezcan en pantalla, es decir, de los distintos elementos que las forman.
- *values*: dentro de ésta hallaremos también archivos en formato XML. Estos archivos determinarán valores de tipo *String*, color o estilo que se utilicen en el programa.
- *menu*[4]: se trata de ficheros XML que contienen los menús de la aplicación.
- *xml*[5]: contiene otros ficheros XML requeridos por la aplicación, como el archivo de preferencias.

[4] Este archivo no tiene por qué estar presente siempre en un proyecto.
[5] Este archivo no tiene por qué estar presente siempre en un proyecto.

Capítulo 4

Aplicación PRESSMATIC

Durante el trascurso de este capítulo se desea mostrar al lector de una manera más detallada cómo se han llevado a cabo los procesos de Análisis, Diseño e Implementación de la aplicación.

4.1. Análisis y diseño de la aplicación

En este apartado se examinarán las funcionalidades que debe presentar la aplicación y, a partir de esto, se establecerán unos requisitos de diseño que darán pie a elaborar el diagrama de casos de uso. Una vez hecho esto, se explicarán algunos detalles del proceso de diseño coordinado de interfaces.

4.1.1. Funcionalidad

Las funcionalidades que aporte la aplicación móvil serán las mismas que se presente en el dispositivo PRESSMATIC a través de la pantalla incorporada, excepto por una serie de características que existirán únicamente en la aplicación Android. Las principales funcionalidades añadidas consisten en habilitar una conexión bluetooth y facilitar un sistema de notificación al usuario mediante el cual conocer el estado de conexión.

PRESSMATIC es un dispositivo electromecánico de asistencia cuyo objetivo es generar un movimiento automático de cierre y pinzado. Para llevar a cabo esta tarea el dispositivo consta de un motor que llevará a cabo los movimientos establecidos mediante electrónica de control. Serán los parámetros del motor los que determinarán de una manera u otra los distintos modos de operación que se implantarán en PRESSMATIC, que serán después traducidos en parámetros de control concretos [1]. Dichos modos de operación se detallan a continuación:

1) El modo paso a paso provoca que el motor genere un pulso controlado de avance o retroceso con cada pulsación de los botones ABRIR o CERRAR respectivamente. El pulso en cuestión será mayor o menor dependiendo del ancho de pulso seleccionado, favoreciendo así un movimiento de pinzado y de agarre de objetos preciso.

2) El modo continuo implementa un movimiento de cierre y apertura constante en el que se controla la frecuencia del motor, traducida en velocidad. Este movimiento será utilizado convenientemente para efectuar acciones con herramientas que

requieran un movimiento continuado, como puede ser realizar un corte con unas tijeras. En este modo el usuario podrá controlar el inicio y la parada de la ejecución mediante el botón de START / STOP.

3) El modo de corte único ejecuta un ciclo programado del motor. Este modo está destinado a desempeñar una acción temporizada que consiste en juntar los cabezales completamente, ejerciendo cierta presión durante unos instantes, para después separarlos y concluir el ciclo. La implementación de este modo sólo incluirá un botón que originará el ciclo descrito.

4) El modo ciclos es un modo inicialmente destinado a llevar a cabo una cantidad determinada de ciclos de apertura y cierre consecutivos posteriores a una señal de confirmación. Sin embargo, este modo no se ha implantado en las herramientas actuales ya que finalmente no ha se ha identificado con ninguna herramienta, aunque no se descarta su uso futuro.

Una vez conocidos los modos de operación, se hace difícil pensar que un usuario que use PRESSMATIC por primera vez relacione dichos modos con la operación que implementan de una manera obvia y directa. El usuario no tiene por qué estar familiarizado con estos modos de implementación y a partir de ahora, sin cambiar nunca la funcionalidad, se procurará adoptar en la aplicación una interfaz intuitiva para el usuario. Mediante dicha interfaz se manejará PRESSMATIC traduciendo los distintos parámetros y modos en algo factible de ser entendido por el usuario.

4.1.2. Requisitos de la aplicación

Los requisitos que tendrá que cumplir la aplicación son muy específicos. Estos dependen principalmente de las necesidades de usuario, en este caso, de controlar el dispositivo PRESSMATIC mediante su dispositivo móvil fácilmente. Es esta necesidad principal la que determinará las exigencias que debe satisfacer el software creado.

Entre los principales requerimientos está crear una aplicación de la que puedan beneficiarse todos, es decir, una aplicación accesible, que a la vez debe ser funcional. Por ello existirán por un lado unos requisitos de accesibilidad y, por otro lado, unos requisitos generales o funcionales que debe cumplir la aplicación para llevar a cabo los objetivos para los que se ha desarrollado.

4.1.2.1. Requisitos funcionales

La aplicación debe integrar una serie de características esenciales para cumplir el objetivo por el que está concebida. El objetivo principal es ser capaz de controlar el dispositivo PRESSMATIC eficazmente, de esta premisa surgen una serie de requisitos:

-Permitir al usuario el control de los distintos modos de operación de una manera clara mediante la manipulación de los parámetros de los mismos, permitiendo a su vez ejecutar las acciones de inicio o parada.

-Realizar una conexión bluetooth robusta y estable con el dispositivo PRESSMATIC. Dicha conexión debe ser persistente entre las distintas actividades, que implementarán los modos de operación de PRESSMATIC, permitiendo una transmisión continúa de información. Además, es estrictamente necesario conocer el estado de la conexión de tal manera que se notifique al usuario si ésta se pierde, si no se ha podido conectar o si se ha realizado con éxito.

-El dispositivo PRESSMATIC debe de estar coordinado con la aplicación**,** es decir, el dispositivo móvil actuará como maestro de la comunicación. En este sentido, todo cambio producido en la pantalla de la aplicación móvil, producirá el mismo cambio en el dispositivo Pressmatic.

4.1.2.2. Requisitos de accesibilidad

Una vez sabido que las compañías proveen a los desarrolladores de distintas pautas de accesibilidad y, a pesar del estado actual de la accesibilidad en las aplicaciones móviles, en esta sección se puntualizan los criterios que se intentarán adaptar en la aplicación para desarrollar una aplicación PRESSMATIC accesible.

Google provee en la página de desarrolladores de Android una serie de publicaciones en las que explica el funcionamiento de sus herramientas de accesibilidad y, por otro lado, aporta información acerca de cómo integrar la accesibilidad de la manera más adecuada en las aplicaciones. Los requisitos que debe cumplir una aplicación Android, según la compañía, para garantizar un nivel mínimo de accesibilidad en una aplicación son los siguientes [18]:

-Identificar los elementos y los controles de la interfaz de usuario que no posean texto visible mediante descripciones significativas y cortas de contenido. Esto se aplica, en particular, a aquellos elementos que contienen imágenes o a casillas de verificación.

-Todos los elementos de la interfaz de usuario que acepten entradas (mediante contacto o escribiendo) deberían poder llegar a utilizarse de una manera alternativa.

-Si se crea una manera personalizada de controlar la interfaz, es decir, a medida para la aplicación, se deberían implementar dichas interfaces de una manera accesible y proveer una descripción de los contenidos de las mismas.

-No usar únicamente el sonido como forma de proporcionar información al usuario. Debe existir siempre un método alternativo para llevar a cabo este proceso, ya que pueden existir usuarios con cierta discapacidad auditiva. Una de estas formas puede ser la vibración.

-Testear la accesibilidad de la aplicación mediante la navegación de la mismas usando métodos alternativos al contacto y habilitando los servicios de accesibilidad que se proveen. La compañía en este caso también pone a disposición del desarrollador las pruebas que debe

superar la aplicación para lograr buenos resultados en niveles de accesibilidad. Estas pruebas incluyen principalmente satisfacer los criterios anteriores, además de verificar una serie de premisas: el usuario debería ser capaz de navegar mediante TalkBack[6] con facilidad, empleando gestos específicos de la aplicación, o utilizando Explore by Touch[7]. Además, los elementos tangibles de la interfaz deberán medir 48 dp (unos 9mm) como mínimo.

Para satisfacer los requisitos anteriores Google proporciona una serie servicios que permiten a todo tipo de usuarios navegar y hacer un uso cómodo de sus aplicaciones, permitiendo también a los desarrolladores crear sus propios servicios. Entre los usuarios para los que se proporcionan este tipo de servicios se incluyen aquellos que padecen limitaciones visuales, físicas, limitaciones propias de la edad o usuarios con problemas de oído. Sin embargo cabe señalar que, como se ha mencionado a la hora de estudiar las distintas plataformas, la documentación de la compañía está **orientada a la diversidad funcional visual.**

Por otro lado, el año 2013 se elaboró desde Ceapat [32] un documento como forma de guía acerca de "Cómo hacer Apps Accesibles", ante la inexistencia de una normativa nacional o internacional con este propósito particular. Dicho documento informa acerca de las necesidades de las personas que poseen una discapacidad y alberga una serie de **recomendaciones** que debe considerar el desarrollador para conseguir una aplicación accesible. Tomando como referencia este documento se determinarán los **requisitos de accesibilidad más relevantes que no hayan sido contemplados anteriormente** (por la compañía Google), para intentarán adaptarlos después a la aplicación. Estos requisitos son los siguientes:

Referidos a las preferencias de usuario

-Crear una interfaz flexible y personalizable que posibilite al usuario cambiar las preferencias de la aplicación mediante la interfaz del sistema. Es importante en este caso que los cambios introducidos en la configuración no necesiten un reinicio del sistema o de la aplicación para surtir efecto.

-Si el sistema precisa de una respuesta en un tiempo establecido, el usuario debería poder ajustar dicho intervalo incluyendo la posibilidad de suprimirlo.

-La interfaz de usuario debe ser acorde con los distintos atributos de visualización, adaptándose a la configuración de color, tamaño, contraste… que haya establecido el usuario.

Referidos a las pautas generales sobre control y uso:

-Se debe permitir al usuario elegir los métodos de entrada y salida preferidos o, al menos, proporcionar métodos alternativos a la pantalla táctil.

[6] Servicio de lectura de pantalla proporcionado por Google. Usa mensajes de voz para describir los resultados de las acciones que se ocurren en la pantalla, como puede ser abrir una aplicación o la aparición de una notificación.

[7] Característica del sistema que funciona con Talkback, permitiendo a los usuarios tocar la pantalla y recibir un mensaje de voz que comunica lo que se encuentra debajo del dedo.

-El número de pasos que el usuario debe realizar para ejecutar una acción debe de ser optimizado, siendo lo deseable que el objetivo se cumpla en dos o tres pasos.

-El usuario debería poder rectificar los efectos de acciones no intencionadas. En el caso de que las mismas no puedan deshacerse, se debe solicitar confirmación antes de realizarlas.

- Las notificaciones deben de ser consistentes y fácilmente identificables si se trata de mensajes del mismo tipo. Esto incluye que siempre aparezcan en la misma posición y posean el mismo formato.

-Tanto los mensajes en general como los mensajes de error deben ser escuetos, sencillos y claros para un usuario no técnico. En el caso de los mensajes de error el sistema debería sugerir soluciones al problema.

Referidos a la compatibilidad con las ayudas técnicas:

-La aplicación debe hacer uso de los servicios de accesibilidad ofrecidos por el sistema operativo en cuestión y debe ser compatible con los mismos de tal manera que no interfiera en la accesibilidad del sistema operativo o de otros productos.

-La aplicación debe proporcionar al usuario acceder a las características de los objetos de la interfaz mediante ayudas técnicas.

-Todos los objetos, elementos, controles e imágenes de la interfaz de usuario deben de estar correctamente etiquetados.

Referidos a las salidas del sistema:

-Evitar presentar elementos que parpadeen, ya que dificultan la legibilidad y pueden causar en algunos usuarios ataques epilépticos.

-Las aplicaciones deben facilitar al usuario la elección de parámetros como el tipo de letra, su tamaño y el color de los controles de la interfaz. Dentro de estas opciones se incluye también la modificación del estilo y la fuente. El objetivo es hacer los textos legibles.

-El escalado de la interfaz debe ser el adecuado de tal manera que el texto y los controles sigan siendo visibles y navegables cuando se alteran mediante el menú de preferencias del sistema.

-El color es una propiedad importante a tener en cuenta, existiendo varios factores a considerar, pudiendo usarse para destacar información pero nunca siendo la única forma de transmitir dicha información. En relación a la interfaz, deben proporcionarse combinaciones de colores preestablecidas y personalizables que hayan sido diseñadas de tal manera que los usuarios con problemas de visión puedan beneficiarse también de la aplicación. Algunos sistemas incluyen la opción de proporcionar esquemas de colores diseñados para personas con discapacidad y proporcionar contraste, generalmente entre primer plano y fondo.
En este caso se entiende por contraste la valoración de la diferencia de aspecto de dos o más partes de un campo que se observan simultáneamente o sucesivamente.

Referidos al sonido:

-Deben ofrecerse funciones que permitan enviar información al usuario del estado actual del dispositivo mediante síntesis de voz y éstos deben originarse justo después del evento que originó dicho estado.

Referidos al soporte al usuario:

-La documentación de la aplicación y la ayuda debe estar redactada en un lenguaje sencillo y directo para favorecer la correcta y eficaz utilización de la misma por los usuarios.

-La información relacionada con las características de accesibilidad implantadas en la aplicación debe estar a disposición de los destinatarios de la misma.

4.1.3. Casos de uso

Ahora que se conocen los requisitos que debe cumplir la aplicación para llevar a cabo su diseño e implementación, se confeccionará el diagrama de casos de uso resultantes de los mismos. Un caso de uso es una sucesión de acciones emprendidas por el sistema, en respuesta a un evento que realiza el actor [33].

El diagrama que se muestra en primer lugar es una herramienta que se aprovecha para sintetizar el comportamiento del sistema y sus actores, que en este caso solo es uno, el usuario.Con la descripción textual posterior de cada caso se presentan tanto el conjunto de condiciones que requiere el usuario, como el software para la utilización de la aplicación.

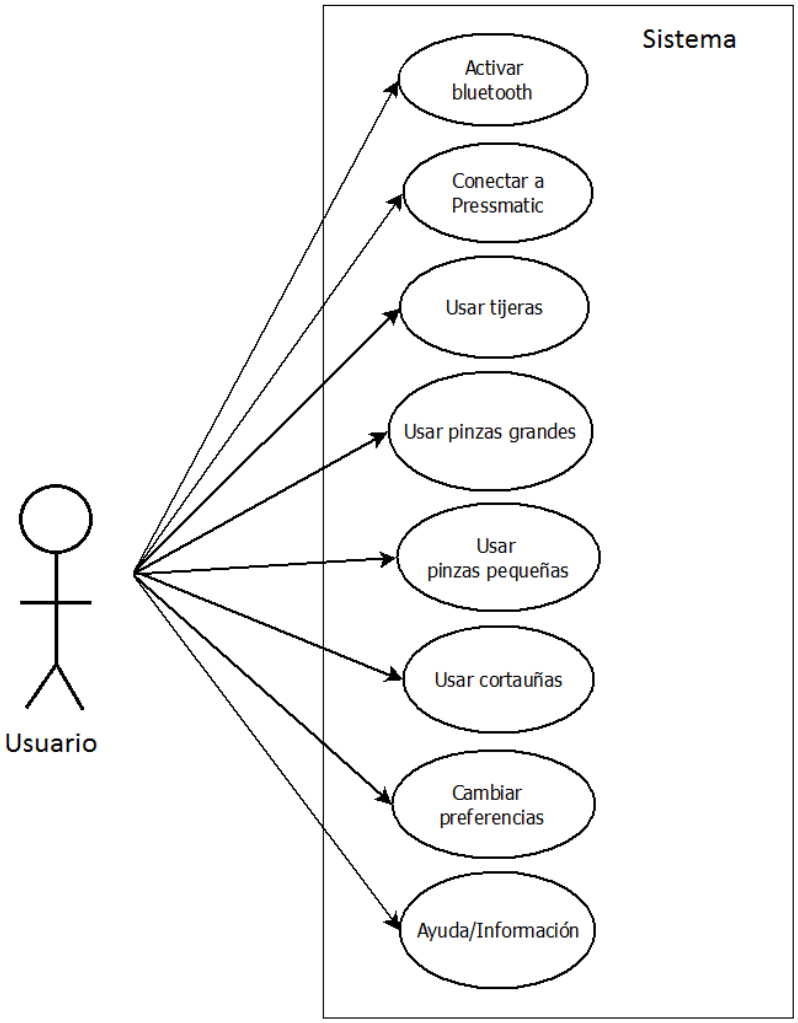

Figura 17. Casos de uso de la aplicación

Para realizar el análisis de dichos casos de uso precisando para cada uno una serie de campos:

-Actores: actores que interactúan con el caso de uso.
-Objetivo: descripción del fin de cada caso de uso.
-Precondiciones: condiciones previas que se deberán verificar para poder llevar a cabo el caso de uso.
-Post-condiciones: condiciones posteriores que se cumplirán una vez se ha haya llevado a cabo el caso de uso en cuestión.
-Secuencia normal: progreso normal del flujo de ejecución del caso de uso.
-Excepciones: secuencia de pasos que se producirán cuando durante la ejecución de una secuencia normal ocurra una excepción.

En los casos de uso se consideran las acciones del sistema, en este caso del dispositivo móvil, en respuesta a una interacción de un usuario externo. Teniendo en cuenta esto, únicamente los sistemas externos que realicen una interacción se considerarán actores. Es por este motivo por el cual el dispositivo PRESSMATIC no se incluye como actor en el diagrama, porque que no interactúa con la aplicación ya que no se comunica con la misma. Sin embargo, sí se produce comunicación en el sentido contrario, de la aplicación hacia el dispositivo electromecánico.

A continuación se especifican cada uno de los casos de uso del sistema:

Caso de uso 1: Activar bluetooth
Actores: Usuario.
Objetivo: Activa la función bluetooth del dispositivo.
Precondiciones: Iniciar la aplicación y no haber activado el bluetooth con anterioridad.
Post-condiciones: El dispositivo estará listo para conectarse a PRESSMATIC.
Secuencia normal:
1. El usuario acepta la activación del bluetooth.
2. El usuario acepta los permisos de la aplicación para usar bluetooth.
3. El bluetooth queda activado.
Excepciones:
Caso a: La activación del bluetooth no se permite.
1a. El usuario no acepta la activación de bluetooth.
2a. La aplicación se cierra mostrando un mensaje de que el bluetooth es necesario.
Caso b: Los permisos para utilizar el bluetooth son denegados.
1b. El usuario acepta la activación del bluetooth.
2b. El usuario no da permisos a la aplicación para usar bluetooth.
3b. La aplicación se cierra mostrando un mensaje de que el bluetooth es necesario.

Caso de uso 2: Conectar a PRESSMATIC
Actores: Usuario.
Objetivo: Conectarse al dispositivo PRESSMATIC.
Precondiciones: El bluetooth del dispositivo debe estar activado y pantalla principal activa.
Post-condiciones: Se podrán utilizar los distintos modos de operación.

Secuencia normal:

1. El usuario pulsa el botón bluetooth.

2. El sistema muestra la lista de dispositivos vinculados y disponibles.

2. El usuario selecciona el dispositivo PRESSMATIC.

3. Tras un breve período de tiempo, se establece la conexión bluetooth emitiendo un mensaje con el dispositivo al que se ha conectado.

Excepciones:

Caso a: Se selecciona un dispositivo PRESSMATIC, pero no está disponible.

1. El usuario pulsa el botón bluetooth.

2. El sistema muestra la lista de dispositivos vinculados y disponibles.

3. El usuario selecciona el dispositivo PRESSMATIC.

4a. Tras un breve período, el sistema emite un mensaje de que no se ha podido conectar.

Caso de uso 3: Usar tijeras

Actores: Usuario.

Objetivo: Utilizar la herramienta de tijeras.

Precondiciones: Conexión bluetooth establecida y pantalla principal activa.

Post-condiciones Se mostrará la pantalla para controlar las tijeras.

Secuencia normal:

1. El usuario pulsa el botón tijeras.

2. El sistema abre la pantalla para usar las tijeras.

3. El usuario selecciona los parámetros de trabajo y pulsa el botón START/STOP según desee.

4. El sistema envía la información correspondiente a PRESSMATIC.

Excepciones:

Caso a: La conexión no se ha establecido.

1. El usuario pulsa el botón tijeras.

2a. El sistema emite un mensaje de que se debe efectuar la conexión primero.

Caso b: Se deshabilita el bluetooth

1. El usuario pulsa el botón tijeras.

2. El sistema abre la pantalla para usar las tijeras.

3b. El usuario desconecta la función bluetooth del dispositivo móvil manualmente.

4b. El sistema, después de unos segundos, emite un mensaje notificando que se ha perdido la conexión.

Caso de uso 4: Usar pinzas grandes.

Actores: Usuario.

Objetivo: Utilizar la herramienta de pinzas grandes.

Precondiciones: Conexión bluetooth establecida y pantalla principal activa.

Post-condiciones: Se mostrará la pantalla para controlar las pinzas grandes.

Secuencia normal:

1. El usuario pulsa el botón pinzas grandes.

2. El sistema abre la pantalla para usar las pinzas grandes.

3. El usuario selecciona los parámetros de trabajo y pulsa el botón ABRIR o CERRAR según desee.

4. El sistema envía la información correspondiente a PRESSMATIC.

Excepciones:

Caso a: La conexión no se ha establecido.

1. El usuario pulsa el botón pinzas grandes.

2a. El sistema emite un mensaje de que se debe efectuar la conexión primero.

Caso b: Se deshabilita el bluetooth.

1. El usuario pulsa el botón pinzas grandes.

2. El sistema abre la pantalla para usar las pinzas grandes.

3b. El usuario desconecta la función bluetooth del dispositivo móvil manualmente.

4b. El sistema, después de unos segundos, emite un mensaje notificando que se ha perdido la conexión.

Caso de uso 5: Usar pinzas pequeñas

Actores: Usuario.

Objetivo: Utilizar la herramienta de pinzas pequeñas.

Precondiciones: Conexión bluetooth establecida y pantalla principal activa.

Post-condiciones: Se mostrará la pantalla para controlar las pinzas pequeñas.

Secuencia normal:

1. El usuario pulsa el botón pinzas pequeñas.

2. El sistema abre la pantalla para usar las pinzas pequeñas.

3. El usuario selecciona los parámetros de trabajo y pulsa el botón ABRIR o CERRAR según desee.

4. El sistema envía la información correspondiente a PRESSMATIC.

Excepciones:

Caso a: La conexión no se ha establecido.

1. El usuario pulsa el botón pinzas pequeñas.

2a. El sistema emite un mensaje de que se debe efectuar la conexión primero.

Caso b: Se deshabilita el bluetooth.

1. El usuario pulsa el botón pinzas pequeñas.

2. El sistema abre la pantalla para usar las pinzas pequeñas.

3b. El usuario desconecta la función bluetooth del dispositivo móvil manualmente.

4b. El sistema, después de unos segundos, emite un mensaje notificando que se ha perdido la conexión.

Caso de uso 6: Usar cortaúñas

Actores: Usuario.

Objetivo: Utilizar la herramienta de cortaúñas.

Precondiciones: Conexión bluetooth establecida y pantalla principal activa.

Post-condiciones: Se mostrará la pantalla para controlar el cortaúñas.

Secuencia normal:

1. El usuario pulsa el botón cortaúñas.

2. El sistema abre la pantalla para usar el cortaúñas.

3. El usuario selecciona el botón CORTAR.

4. El sistema envía la información correspondiente a PRESSMATIC.

Excepciones:

Caso a: La conexión no se ha establecido.

1. El usuario pulsa el botón cortaúñas.

2a. El sistema emite un mensaje de que se debe efectuar la conexión primero.

Caso b: Se deshabilita el bluetooth.

1. El usuario pulsa el botón cortaúñas.

2. El sistema abre la pantalla para usar el cortaúñas.

3b. El usuario desconecta la función bluetooth del dispositivo móvil manualmente.

4b. El sistema, después de unos segundos, emite un mensaje notificando que se ha perdido la conexión.

Caso de uso 7: Cambiar preferencias

Actores: Usuario.

Objetivo: Modificar las preferencias de usuario.

Precondiciones: Bluetooth activado y pantalla principal activa.

Post-condiciones: La interfaz de usuario podrá verse cambiada.

Secuencia normal:

1. El usuario pulsa el botón de preferencias.

2. El sistema muestra la pantalla de preferencias.

3. El usuario selecciona la parte de la interfaz de usuario que desea modificar.

4. El sistema muestra los parámetros de elección posibles.

5. El usuario selecciona el parámetro que desee.

6. El sistema modifica la interfaz según los parámetros seleccionados.

Caso de uso 8: Ayuda/Información

Actores: Usuario

Objetivo: Mostrar al usuario la pantalla de ayuda e información de la aplicación.

Precondiciones: Bluetooth activado y pantalla principal activa.

Post-condiciones: Se mostrará la pantalla de ayuda e información.

Secuencia normal:

1. El usuario pulsa el botón de ayuda/información.

2. El sistema muestra la pantalla de ayuda/información.

4.1.4. Coordinación de creación de interfaces

El proceso de desarrollo de la aplicación como tal se trata de un trabajo individual, de todos modos, el diseño de la interfaz se realizará de manera coordinada. La coordinación de diseño se efectúa entre Rodrigo Martín [4], encargado de implantar el control PRESSMATIC desde la pantalla del propio dispositivo, y el autor del presente proyecto.

Los elementos que se ponen en común al llevar a cabo la coordinación entre dichos proyectos son los distintos elementos que formarán parte de la interfaz de usuario de ambas partes. Estos elementos son principalmente las múltiples actividades que formarán parte de

la aplicación, la apariencia de las mismas y los botones que las forman, que estarán en consonancia entre sí.

El objetivo al ejecutar esta tarea es principalmente permitir a la persona que use PRESSMATIC, un uso parejo del dispositivo PRESSMATIC y de la aplicación móvil con el mismo nombre. Este afán da lugar a la creación de un diseño conjunto de los botones o iconos de PRESSMATIC.

4.1.5. Diseño de la interfaz. Iconos de Pressmatic.

El diseño de los iconos existentes, tanto en la pantalla de PRESSMATIC como en la aplicación móvil, ha sido dirigido principalmente por Alonso Rosado mediante la herramienta Inkscape [34]. Este trabajo ha sido siempre supervisado por Rodrigo Martín y el autor del presente proyecto, diseñadores de la interfaz de la pantalla de PRESSMATIC y de la aplicación, de tal manera que se verificara el cumplimiento de los requisitos de accesibilidad vigentes.

Unas de las pautas de accesibilidad más consideradas son aquellas referidas a las salidas del sistema y, entre ellas, el color es una propiedad a tener en cuenta. En el diseño de los iconos esto es especialmente importante, ya que estos son una de las principales formas de comunicación con el usuario. En cuanto al color, es fundamental la existencia de contraste entre el fondo de la pantalla y los elementos existentes en la misma, tal y como se menciona en el apdo.4.1.2.2, de tal manera que cualquier usuario pueda beneficiarse del diseño. Conocido el concepto de contraste, explicado anteriormente, se contempla el uso de colores complementarios en la interfaz que, de forma natural, originan un alto contraste entre ellos. Para esto se tomará como referencia el círculo cromático de la figura 18.

Figura 18. Circulo cromático [35]

Por otro lado, además de mediante el empleo de colores complementarios, un detalle a tener en cuenta es que el contraste puede ser también ocasionado por una considerable subida o bajada de brillo.

Estas consideraciones dan lugar a la creación de una serie de iconos que formarán parte de la interfaz tanto del dispositivo como de la aplicación PRESSMATIC. Dichos iconos han sido diseñados con un contraste de brillo interno y con la intención de utilizarlos sobre un fondo que, a su vez, contraste con el propio botón. Por este motivo se decide crear dos sets completos de iconos para la interfaz, basándose en un fondo rojo y otro azul, que harán la función de botones en la interfaz. En las figuras siguientes se muestran algunos de los iconos diseñados:

Figura 19. Botones de bluetooth y botones de tijeras

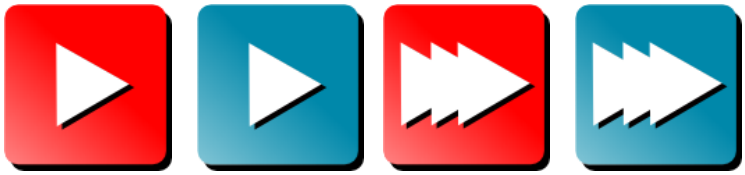

Figura 20. Botones de velocidad

Figura 21. Botones START / STOP

Asimismo, se apreciará en interfaz la creación de iconos para botones de mayor anchura, como en la figura 21, de hasta tres veces más que el icono estándar. Esta decisión de diseño se implanta en los botones cuya utilización por parte del usuario esté destinada a realizarse de una manera mucho más frecuente.

No obstante, se han tenido en cuenta otros rasgos a parte del color. Entre estos destaca expresar la funcionalidad de los botones de una manera clara y, en la medida de lo posible, incluir un etiquetado adecuado de los botones. En relación a esto, los botones contienen o bien un etiquetado o bien una figura que expresa su función. El diseño fue concebido así debido a que se estimó que la interfaz era suficientemente clara si se realizaba un diseño apropiado de los botones. Se concluyó que, si se realizaba un etiquetado de todos y cada uno de los botones, en particular de los que ya poseen una imagen, la interfaz se podría ver sobrecargada de información. Puesto que se deseaba diseñar una interfaz clara y directa, esto no era deseable.

Los botones han sido creados de tal manera que el usuario asocie de manera inmediata los mismos con la función que desempeñan. Aun así, si el usuario encontrara poco claras las funciones de dichos botones, siempre podría acudir a la pantalla de ayuda localizada en la pantalla principal para solventar las dudas que pudieran surgir.

Por último, cabe mencionar que se han diseñado otros iconos que formarán parte de la aplicación pero que no se encuentran dentro de los conjuntos mencionados previamente. Entre ellos se encuentra el icono que indica que existe una conexión activa y el icono de la aplicación:

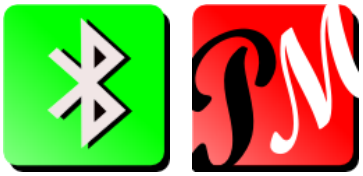

Figura 22. Botón bluetooth activo e icono de la aplicación

4.2. Implementación

En este apartado se explicará con detalle todo lo necesario para entender el funcionamiento de la aplicación desde el protocolo bluetooth hasta la programación de las distintas actividades.

4.2.1. Diagrama de clases

En la página siguiente se adjunta el diagrama de clases con cada uno de los métodos pertenecientes a las clases que componen el proyecto. De esta manera se pretende que el lector tenga una idea global de las clases que forman parte de la aplicación.

Como se observará, tanto la actividad principal (Pressmatic) como las actividades relativas a los modos de operación están directamente asociadas al servicio BluetoothConexion. Esto es debido a que cada una de estas actividades contiene una instancia de esta clase, que es utilizada posteriormente para almacenar la información del servicio.

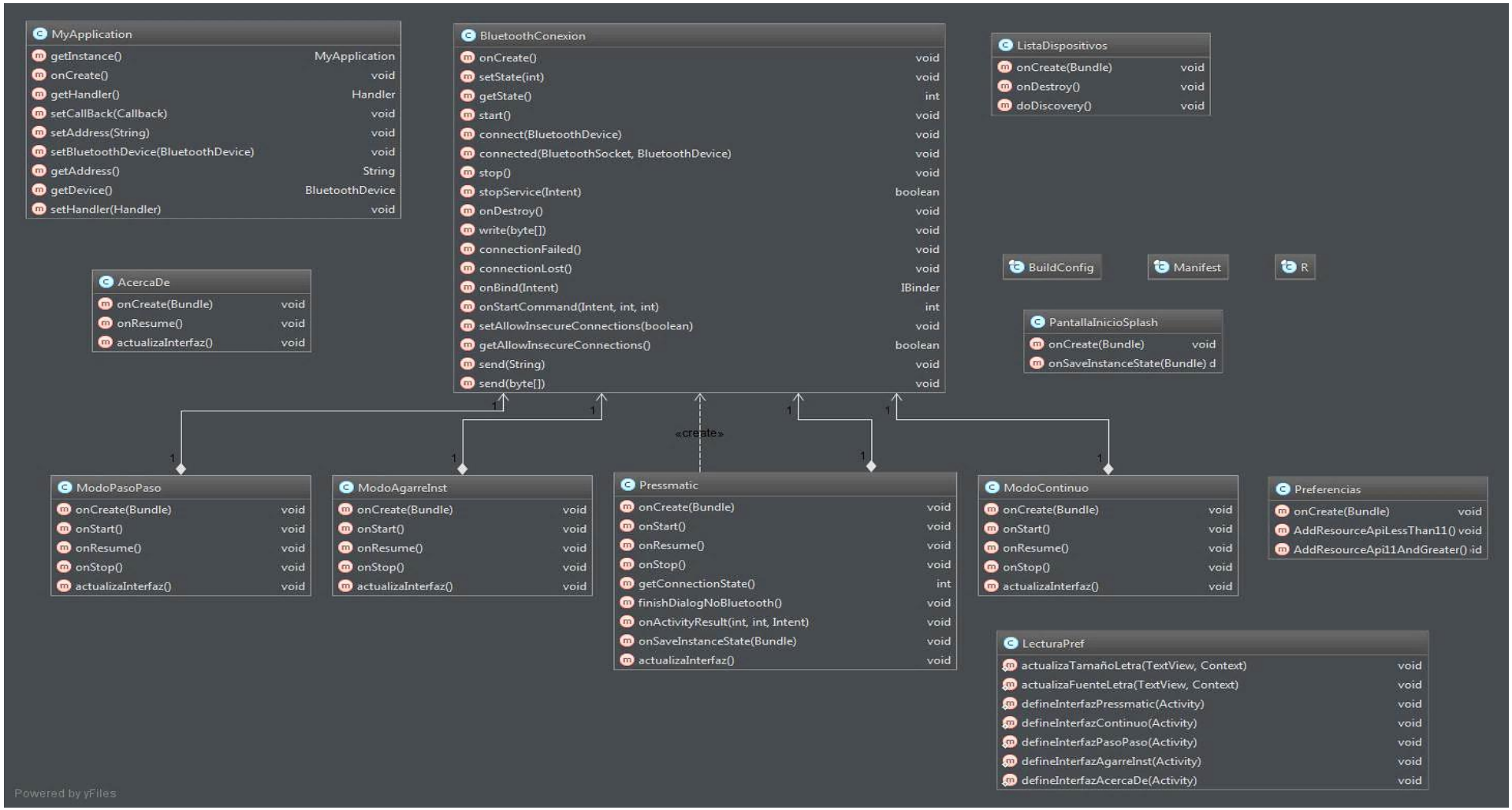

Figura 23. Diagrama de clases

4.2.2. Protocolo bluetooth

En esta sección se pretenden esclarecer los motivos que han llevado a crear la conexión bluetooth con el dispositivo PRESSMATIC de una manera determinada y se desean definir las principales características que definen dicha conexión. Como se menciona previamente, la implementación del protocolo bluetooth en la aplicación ha sido llevada a cabo con la ayuda de Alonso Rosado. Esto ha sido posible después de realizar una serie de pruebas con el módulo bluetooth HC-05, módulo seleccionado minuciosamente por esta persona.

Cabe señalar que ha sido el protocolo bluetooth el que ha planteado más dificultades a la hora de ser implementado con efectividad. Inicialmente, la manera de establecer la conexión con el dispositivo PRESSMATIC estaba basada en el código abierto proporcionado por una aplicación denominada "Blueterm" [36] y un ejemplo existente en la página de desarrolladores de Android bajo el nombre "BluetoothChat". Ambas aplicaciones establecen su conexión únicamente en una actividad, sin la necesidad de extender y mantener esta conexión en otras actividades. No obstante, la aplicación PRESSMATIC requiere una conexión constante y estable entre actividades, permitiendo la navegación entre los modos de operación a utilizar.

Se barajaron varias opciones para efectuar la conexión con éxito: crear una interfaz que solo dispusiera de una actividad, estableciendo así un protocolo conocido, iniciar la conexión en la actividad principal para después restaurarla en la siguiente, y, por último, crear la conexión mediante la utilización de un *Service*. La primera planteó problemas técnicos, ya que era imposible considerar la posibilidad de desarrollar una única pantalla donde se llevara a cabo todo el control de PRESSMATIC y la segunda de estas opciones resultó ser extremadamente ineficiente por la cantidad de tiempo que requiere volver a establecer la conexión. Fue, por lo tanto, la tercera opción, el *Service*, la que ofreció más facilidades a la hora de crear la conexión, ya que, como se ha explicado en el apartado 3.4, permite ejecutar operaciones de mayor duración en segundo plano, posibilitando una conexión estable entre actividades.

Un servicio o *Service* se trata de un objeto con un ciclo de vida definido [18]. Éste se puede iniciar de dos maneras, mediante el método `onStartCommand()` o por el método `onBind()`, lo cual influye en el ciclo de vida del servicio. La principal diferencia entre dichos métodos estriba en que si se emplea `onBind()` el servicio y con ello, la conexión, se asocian a la actividad donde se ha producido la llamada a dicho método. De esta manera si se destruyen todas las actividades ligadas el servicio se ve finalizado. Por otro lado, usando `onStartCommand()` no se suspende la conexión a pesar de destruir la actividad, siendo el usuario el responsable de llevar a cabo esta tarea.

Para habilitar un protocolo estable de comunicación bluetooth entre dos dispositivos es imprescindible seguir una serie de pasos pero, en primer lugar, es necesario comprender el uso de una serie de objetos que la API de Google pone a disposición de los desarrolladores. Los principales objetos de los que hace falta tener conocimiento son [37]:

1. Thread: se trata de un hilo de ejecución, es de tipo Runnable. Mediante un hilo de ejecución se puede iniciar un proceso en un segundo plano de la aplicación. Se pueden crear tantos thread como se desee teniendo en cuenta que funcionan de manera independiente a la aplicación, sin capacidad para alterar el hilo principal de la aplicación. Mediante el uso de un thread se establecerá la conexión bluetooth.
2. Handler: permite la gestión de mensajes y objetos de tipo Runnable asociados con los mensajes procedentes de un thread. Se trata de un puente entre un hilo secundario y el hilo principal de la aplicación, ya que permite enviar mensajes a éste último.
3. BluetoothSocket: un socket, por lo general, provee de un protocolo de control de transmisión haciendo posible el intercambio de datos entre dispositivos. En este caso posibilita dicho intercambio haciendo uso del adaptador bluetooth que posee el dispositivo móvil**,** habilitando la transmisión de información al módulo existente en el dispositivo PRESSMATIC.
4. Bundle: se trata de un objeto mediante el cual se envían datos de distintos tipos entre las actividades de Android. Este método es llamado, por ejemplo, cuando se inicia una actividad.

Para establecer la conexión exitosamente se hace imprescindible que el dispositivo disponga de un módulo bluetooth. En general, todos los dispositivos que salen al mercado hoy en día lo poseen, pero hace falta comprobarlo para asegurarse de ello. Esto se hace mediante el siguiente fragmento de código:

```
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
           if (mBluetoothAdapter == null) {
           finishDialogNoBluetooth();
                return;
           }
```

El objeto BluetoothAdapter permite realizar tareas fundamentales en la conexión, como puede ser iniciar la búsqueda de dispositivos y mostrar aquellos emparejados, instanciar un objeto BluetoothDevice (dispositivo bluetooth) o crear un socket del tipo BluetoothServerSocket. En el supuesto caso de que no exista un adaptador estándar se devolverá al usuario un mensaje informándole de que la conexión es imposible. En caso afirmativo se pedirá al usuario que active el bluetooth y de permisos a la aplicación para usarlo.

Figura 24. Solicitud de activación bluetooth y solicitud de permisos bluetooth

Cuando la función bluetooth del dispositivo móvil esté activada el usuario se encontrará en la pantalla principal de PRESSMATIC, desde donde se podrá establecer la conexión pulsando en primer lugar el botón de bluetooth.

El proceso de creación del servicio se efectuará, como es lógico, no antes de que se conozca el dispositivo a conectar. Para ello se muestran los dispositivos vinculados y no vinculados disponibles en la actividad "ListaDispositivos". Una vez se seleccione el dispositivo a conectar en la lista, se crea un objeto del tipo BluetoothDevice. Este objeto provee la dirección MAC del módulo receptor necesaria para inicializar la inminente conexión. Finalmente, se llama al método `startService(Intent intent)` que realizará una llamada al método `onStartCommand()` de la clase BluetoothConexion. De esta forma el servicio se verá creado y se creará un hilo secundario de ejecución encargado de la conexión. En este hilo se llamará a la función `connect()`, principal responsable de crear lo necesario para posibilitar definitivamente la comunicación.

El objeto `mBluetoothConexión` (objeto de la clase BluetoothConexion) será el ocupado de contener el servicio mencionado, facilitar el contexto de la aplicación y el **handler** de **comunicación** con el **hilo principal** de la aplicación. Para que esto sea posible se recurre a la clase "MyApplication", que hereda sus propiedades de la clase Android Application [18], una clase base que permite mantener un estado global de la aplicación. De este modo se posibilita, mediante una serie de métodos, acceder a variables globales desde cualquier punto de la aplicación.

Es digno de mencionar que en cada actividad se hace necesario el uso de los métodos `bindService()` y `unBindService()` en determinados momentos. El primero de ellos permitirá al servicio crear una copia en cada actividad del objeto BluetoothConexion creado previamente, vinculando la actividad, y el segundo de estos métodos desvinculará la actividad en cuestión del servicio.

Para gestionar la conexión y conocer su estado en todo momento se empleado una máquina de estados [36] que proporcionará **información** a la aplicación **de manera interna**. En dicha máquina de estados podemos encontrar tres situaciones distintas: `STATE_NONE`, `STATE_CONNECTING` y `STATE_CONNECTED`, designadas por un número entero llamado `mState`. Cada uno de los distintos estados dará lugar a una serie de hilos de conexión que en primer lugar intentarán establecer una conexión desde cero (`STATE_NONE`). Posteriormente, usando el método `connect()`, se cancela todo hilo intentando conectar para centrarse en la conexión en curso. Al llamar dicho método se crea un nuevo ConnectThread, que gestiona los hilos de conexión del dispositivo y declara dos objetos, InputStream y OutputStream, pilas de lectura y escritura de datos respectivamente. Después de esto, se actualiza el estado a `STATE_CONNECTING`. El último paso a realizar consiste en habilitar el socket de comunicación entre los dispositivos el cual administra la conexión y desempeña las transmisiones. Esto se realiza mediante el método `connected()`, que inicia el ConnectThread creado con anterioridad y fija el estado a `STATE_CONNECTED`.

Por otra parte, y como es de esperar, la elección del dispositivo de la lista que provee la aplicación no asegura establecer la conexión y tampoco asegura que ésta no se pierda. Esto se debe a que la conexión bluetooth puede verse condicionada por diversos factores, como que el módulo destinatario esté conectado y disponible, que se encuentre al alcance del dispositivo móvil o simplemente se puede producir un fallo a la hora de conectar debido a elegir un dispositivo incorrecto. Llegado el caso de que se produjera alguno de estos eventos, se realizará una llamada a uno o varios de los siguientes métodos según corresponda: `stop()`, `stopService()`, `connectionLost()` y `connectionFailed()`.

Con la finalidad de mantener al usuario informado por la interfaz de pantalla del estado de la conexión, se crea un Bundle destinado al envío de mensajes “Toast”. Dichos mensajes se muestran en la interfaz de usuario durante un periodo de tiempo establecido y desaparecen después de dicho intervalo formando, junto con los textos, la base de la **comunicación** de la **aplicación con el usuario**.

Figura 25. Ejemplo de mensaje Toast

El proceso de conexión se desarrolla de la manera descrita, puesto que los métodos utilizados como `connect()`, `connected()`, `stop()`… poseen una propiedad característica, son `syncronized`. Esta característica favorece que solo se pueda ejecutar un método de este tipo sobre un mismo objeto de manera simultánea, causando que el proceso se desarrolle adecuadamente y en el orden deseado [17].

Lo último que falta por abordar para conocer de manera general como funciona la conexión bluetooth es el envío de información desde la aplicación al dispositivo PRESSMATIC. Para cumplir el envío de datos de manera adecuada, se convierte en una tarea necesaria tener un conocimiento del número de bytes que se enviarán. Sabido esto, el envío de información se realizará utilizando el método `send()`, cuyo código se adjunta a continuación:

```
public void send(String data) {
        send(data.getBytes());
        }
public void send(byte[] data) {
         write(data);
        }
```

Este método convierte, mediante el uso de dos métodos, el *String* a enviar en un conjunto de bytes de salida que se enviarán al módulo destinatario llamando al método `write()`. Este método, a su vez, lo enviará a la pila de datos de salida (OutputStream). Por lo tanto, valiéndose del método `send()`es posible enviar datos de manera asíncrona a PRESSMATIC y así darle a conocer, por ejemplo, el modo que se ha seleccionado o la tarea que se quiere realizar en cada momento.

En base al envío de datos se establecen una serie de comandos con el fin de registrar la navegación entre los modos seleccionados y la selección de los parámetros de trabajo. De

esta manera, la comunicación del usuario con Pressmatic tanto mediante la aplicación móvil como mediante la pantalla táctil del dispositivo PRESSMATIC se unificaría, facilitando la transición entre pantallas y la comunicación con el sistema mecánico. Esto es así ya que los comandos que recibirá la electrónica de control del dispositivo PRESSMATIC por ambas partes serán iguales.

Por otro lado, el proceso abrirá la puerta al envío de datos a la pantalla táctil permitiendo la actualización del estado de la misma cuando se reciba un comando desde el móvil por comunicación serial. Estos comandos junto con los modos que van asociados los mismos se muestran en la tabla 2:

Grupo	Control	Comando
Menú de selección	Modo Continuo	b
	Modo Paso a Paso	c
	Modo Cortauñas	e
	Modo Ciclos	d
	Bluetooth	f
Modo Continuo	Lento	5
	Medio	6
	Rápido	7
	Start/Stop	0
Modo Cortaúñas	Cortar Una	8
Modo Paso a Paso	Abrir	1
	Cerrar	2
Bluetooth	Desconectar	9
Común a todos los modos	Atrás/Inicio	a
Modo Ciclos (actualmente no incluido en la interfaz)	Sumar un Ciclo	3
	Iniciar proceso de apertura y cierre	4

Tabla 2. Comandos enviados a PRESSMATIC

Por último, es necesario indicar la existencia de dos booleanos que permitirán conocer de manera más precisa que es lo que ocurre con la conexión, y así poder gestionarla mejor:

- `mFirstConnection` es creado para evitar que se llame al método `onBind()` al volver a la actividad principal cuando no se ha creado el servicio. Si la llamada a este método se hiciera antes de existir el servicio la aplicación incurriría en fallos.
- `mBound` creado para evitar destruir el servicio cuando éste no se ha creado, lo cual provocaría fallos.

4.2.3. Pantalla de presentación

Esta actividad es la encargada de iniciar la aplicación y su objetivo es presentarla al usuario mostrando el nombre y los organismos implicados en la creación de la misma. Esta pantalla aparecerá durante un espacio de tiempo determinado, que en este caso son 1,5 segundos (1500 milisegundos). Este tipo de actividades temporales que aparecen al inicio de una aplicación se denominan "Splash".

Figura 26. Pantalla Splash

Para llevar a cabo lo descrito se hace uso de un temporizador de Android, un objeto Timer, que mediante un TimerTask asigna el tiempo de duración de la actividad (`duracionSplash`), establece la actividad que aparecerá luego mediante un Intent (ver apdo. 3.4) y finaliza la actividad actual después utilizando el método `finish()`. De esta manera la pantalla de presentación se destruye y no vuelve a aparecer hasta reiniciar la aplicación. El código correspondiente a la clase que desempeña esta tarea se muestra a continuación:

```
public class PantallaInicioSplash extends Activity {
    //Duración de la pantalla inicial en ms
    private long duracionSplash = 1500;
    boolean usage = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.splashscreen);
      //Creación de la tarea a realizar durante el tiempo establecido
        TimerTask task = new TimerTask() {
                @Override
                public void run() {
                    if(!usage){
                        Intent InicioMain = new Intent
                         (PantallaInicioSplash.this,Pressmatic.class);
                        startActivity(InicioMain);
                        usage=true;
                        PantallaInicioSplash.this.finish();
                    }
                }
        };
      //Temporizador al que se le pasan la tarea y la duración
        Timer temporizador = new Timer();
        temporizador.schedule(task,duracionSplash);

    }
}
```

Es preciso subrayar que en la creación de cualquier actividad se debe especificar siempre el contenido de la misma, es decir, la interfaz de usuario a la que va a asociada la actividad o, lo que es lo mismo, el contenido que se quiere mostrar en pantalla. Para ello se utiliza el método `setContentView(View view)`, al que se le pasa un parámetro del tipo View, en general, un archivo tipo Layout cuya extensión es XML. Lo que hace este método, en definitiva, es definir la ruta a dicho archivo.

En este caso es al archivo layout que determina los elementos de pantalla se encuentra en *Pressmatic/res/layout/splashscreen.xml*. El código correspondiente a dicho archivo consiste simplemente en un *RelativeLayout* que contiene tres elementos: un *TextView* y dos *ImageView*. El primero define el mensaje de bienvenida y los *ImageView* muestran imágenes, en este caso, el logo de la universidad Carlos III y de la fundación Universia.

4.2.4. Pantalla principal

La pantalla que se detalla ahora es la primera actividad en la cual el usuario podrá interactuar con la aplicación de manera directa. Se trata de actividad principal en la que se llevan a cabo las acciones más importantes para el posterior manejo de la aplicación PRESSMATIC. En la pantalla, visible en la figura 27, se ha intentado diferenciar tres áreas a la hora de disponer la interfaz, en concordancia con otras aplicaciones existentes, que son:

1. Barra de información y preferencias de usuario: en esta sección al usuario se le muestra el nombre de la aplicación y dos botones. El botón situado a la izquierda del nombre posee un icono de información llevará al usuario a la sección de ayuda al pulsarlo y al otro lado se encuentra un icono que trasladará al usuario al menú de preferencias, donde podrá configurar la aplicación PRESSMATIC como guste.
2. Zona de instrucciones: en esta parte de la pantalla se hace una descripción textual breve de las acciones que debe seguir el usuario para comenzar a utilizar la aplicación.
3. Área de acción del usuario: en esta zona se inicializa la conexión y se escoge el modo de operación que se desea llevar a cabo mediante una serie de botones.

Figura 27. Pantalla principal de Pressmatic

La disposición de los elementos de esta pantalla se ha establecido mediante una vista *LinearLayout* que los engloba. Dentro de esta distribución vertical tanto el área de la barra informativa, como el área de acción del usuario implementan cada una un *RelativeLayout* dentro de la vista principal. Este tipo de disposición favorece la colocación de los distintos elementos permitiendo colocarlos de manera relativa entre sí.

A la hora de construir la interfaz y con el fin de hacer compatible la aplicación con una amplia variedad de dispositivos, independientemente del tamaño de la pantalla, se hace necesario determinar en cada pantalla el peso que va a ocupar cada parte. Esto implica asignar un espacio relativo a cada componente sobre el espacio total de la pantalla mediante el atributo `layout_weight` (ver apdo 3.3).

Se hace necesario especificar que en cada una de las actividades, exceptuando aquellas que no pueden ser modificadas por el usuario como el menú de preferencias o la lista de dispositivos, se realiza un refresco de los elementos de la interfaz con motivo de realizar una **lectura de las preferencias de usuario**, que se detallarán más adelante. Este refresco se efectúa cuando la actividad pasa por el estado *onResume()*. Esto se efectúa en dos fases:

- Por una parte, se actualizarán los distintos elementos que forman la interfaz como son el texto y la apariencia de la pantalla. Estas modificaciones son efectuadas mediante el uso de métodos de una clase pública llamada *"LecturaPref"* especialmente desarrollada para llevar a cabo esta función, que lee las preferencias de usuario. La implementación de esta clase se detallará más tarde.

- Justo después se reestablecen los listeners[8] de los botones. La actualización de los mismos es totalmente imprescindible de forma posterior a la modificación de la interfaz, ya que se trata de una interfaz "nueva". De esta manera, se permite al usuario utilizar PRESSMATIC sin la necesidad de precisar reiniciar la aplicación para asimilar los cambios. Para ejecutar esta acción se crea un método denominado `actualizaInterfaz()`,que contiene las llamadas a los listeners de los botones y está presente en cada actividad pero es distinto según los elementos de cada una. Si este método no se llamara, la vista de la pantalla se vería modificada pero los botones no llevarían asignados ninguna funcionalidad.

Como se indica en la imagen de la actividad, la primera acción que el usuario debe llevar a cabo es conectarse al dispositivo PRESSMATIC. Para ello se pulsará el botón con el símbolo de bluetooth, denominado *BluetoothButton*, que se trata de un *ToggleButton*. Este tipo de botón cambia entre estado activo (estado `isChecked`) e inactivo y desempeña funciones distintas en función de su implementación. El listener de este botón, que se muestra ahora, será distinto a los que se verán posteriormente.

[8] Un listener es un capturador de eventos de la interfaz de usuario que permite, mediante su implementación en los distintos elementos de tipo *View*(vistas), atender a eventos de distintos tipos y actuar en consecuencia. En general se usará un listener cuyo evento sea `onClick()`, para elementos como los botones , al que se llamará cuando el usuario toque el elemento en cuestión. No obstante, existen otro tipo de eventos que se pueden capturar a parte del simple contacto, como pueden ser `onLongClick()` que detecta una pulsación larga o `onFocusChange()` que detecta el cambio de foco.

```
BluetoothButton.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView,
            boolean isChecked) {
            //Acción que se ejecuta cuando se pulsa el botón en primer
            lugar, para conectar.
                if (isChecked){
                    if (!mBound && mFirstConnection){
                  Intent serverIntent = new Intent(PRESSMATIC.this,
                  ListaDispositivos.class);
                  startActivityForResult(serverIntent,
                  REQUEST_CONNECT_DEVICE);}
                } else {
            //Acción que se ejecuta cuando el botón se pulsa después
            de establecer la conexión, para desconectar.
                    if(mBound){
                        unbindService(mConnection);
                        mBluetoothConexion.stop();
                        mBluetoothConexion.start();
                        mFirstConnection=true;}
                }
            }
        });
```

Al pulsar el botón cambiará a activo y automáticamente aparecerá otra actividad, en forma de diálogo, que muestra la lista de dispositivos vinculados y proporciona la opción de buscar otros dispositivos disponibles.

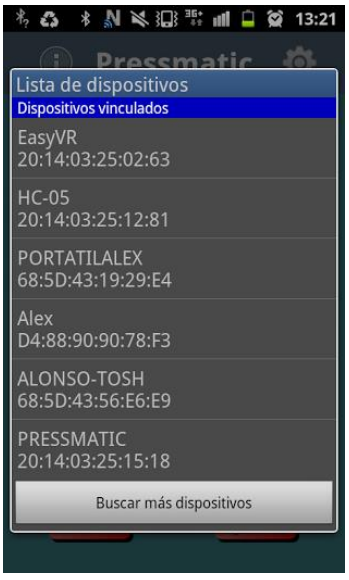

Figura 28. Lista de dispositivos

Esta actividad surge de hacer la llamada al método `startActivityForResult` del botón, al que se le pasan dos parámetros, un Intent y un entero. Este método inicia una actividad de la cual se quiere obtener algo específico, en este caso, inicia la actividad

ListaDispositivos de la cual se desea obtener como *"Result"* un dispositivo. Una vez dicha actividad finaliza (al escoger un dispositivo o volver atrás), se produce una llamada a al método `onActivityResult()`. Este método es el encargado de determinar lo que ocurre en este caso (`REQUEST_CONNECT_DEVICE`) y en otra situación (`REQUEST_ENABLE_BT`), cierra la aplicación en el caso de que el usuario no se permita el uso del bluetooth. El código correspondiente a este método es el siguiente:

```
public void onActivityResult(int requestCode, int resultCode, Intent
data) {
    switch (requestCode) {
        case REQUEST_CONNECT_DEVICE:

            // When DeviceListActivity returns with a device to connect
            if (resultCode == Activity.RESULT_OK) {
              // Get the device MAC address
              String address = data. getExtras()
              .getString(ListaDispositivos.EXTRA_DEVICE_ADDRESS);
              // Get the BLuetoothDevice object
              BluetoothDevice device =
              BluetoothAdapter.getRemoteDevice(address);
              MyApplication mApplication =
              (MyApplication)getApplicationContext();
              mApplication.setAddress(address);
              mApplication.setBluetoothDevice(device);
              // Attempt to connect to the device VIA SERVICE
              startService(new Intent
              (PRESSMATIC.this,BluetoothConexion.class));
              Intent intent = new Intent(this,
              BluetoothConexion.class);
              bindService(intent,mConnection,
              Context.BIND_AUTO_CREATE);
              mFirstConnection = false;
            // When DeviceListActivity does NOT returns with a device
            } else if (resultCode == Activity.RESULT_CANCELED){
              EstadoBoton = false;
              BluetoothButton.setChecked(EstadoBoton);
            }
              break;

        case REQUEST_ENABLE_BT:

            // When the request to enable Bluetooth returns
            if (resultCode != Activity.RESULT_OK) {
                    finishDialogNoBluetooth();
                }
        }
    }
```

En este caso, si se devuelve un dispositivo, se intentará llevar la creación del servicio de conexión. Si se realiza con éxito se cambiará el estado del botón de bluetooth a verde indicando éxito y aparecerá un mensaje Toast. En el caso de que cancele la conexión no se pueda establecer por algún motivo se emitirá un mensaje y se volverá al estado inicial (Figura 29).

Figura 29. Conexión realizada con éxito y conexión fallida

Cuando el botón se encuentra en el estado, conectado si se realiza de nuevo una pulsación del mismo, se provocará la llamada al método `unbindService(mConnection)` y a los métodos `stop()` y `start()` del objeto mBluetoothConexion, provocando así que el servicio se vea destruido y generando un mensaje de que pérdida de conexión.

Para que el usuario use la aplicación de manera ordenada y se favorezca la navegación por los distintos modos de operación, se ha establecido que el acceso a los mismos sea posible únicamente cuando se establezca la conexión a PRESSMATIC y nunca antes de esto. Esto se puede apreciar en la implementación de los listeners de los botones y por ello se muestra el código correspondiente a uno de ellos:

```
Corte.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
       // Este if se cumple solo si la conexión se ha establecido
        if(mBluetoothConexion.getState()== BluetoothConexion.STATE_CONNECTED){
              //Se manda el comando correspondiente al cambio de pantalla
              String corte = "b";
             mBluetoothConexion.send(corte);
              Intent InicioCorte = new Intent (
              PRESSMATIC.this,ModoContinuo.class);
              startActivity(InicioCorte);
        } else {
       //De lo contrario se manda un mensaje al usuario
              Toast.makeText(getApplicationContext(),getString(R.String.toast_c
              onnect_first), Toast.LENGTH_SHORT).show();}
        }
});
```

En el listener de este botón (Corte), como en el del resto de botones de los modos de operación, existe esta condición que solo será cumplida si el objeto mBluetoothConexion se encuentra en estado conectado. Si esto es así se procederá a enviar un carácter en forma de *String* a PRESSMATIC mediante el método `send()`, informándole del cambio de actividad, y se iniciará la actividad correspondiente con un Intent. Por el contrario, si el usuario intentara acceder a dichos modos y no existiera dicha conexión, le aparecería un mensaje emergente indicando que debe realizar la conexión en primer lugar, como se aprecia en la figura 30.

Figura 30. Ejemplo de instrucción al usuario

Es digno de mencionar que los listeners de los distintos botones de los cuatro modos de operación, se diferencian únicamente en dos facetas: el carácter que envían, que dependerá del caso en cuestión (ver Tabla 2), y la actividad a la que trasladan al usuario al pulsar dichos botones. Por su parte, los botones de información y preferencias tienen una implementación más sencilla:

```
Info.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
            // Cambio de actividad debido al Intent
            Intent Informacion = new Intent
            (PRESSMATIC.this,AcercaDe.class);
            startActivity(Informacion);
            }
        });
Settings.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
            // Cambio de actividad debido al Intent
            Intent SettingsMenu = new Intent
            (PRESSMATIC.this,Preferencias.class);
            startActivity(SettingsMenu);
            }
        });
```

El último hecho a destacar de esta actividad es que, para lograr la coordinación entre la pantalla del dispositivo PRESSMATIC y la pantalla del dispositivo Android, se realiza en la llamada a *onResume()* el envío del carácter correspondiente a esta pantalla, la pantalla principal de PRESSMATIC. Gracias a este fragmento de código y, si la conexión ha sido ya establecida, se enviará una notificación al dispositivo PRESSMATIC cada vez que el usuario vuelva a esta pantalla, permitiendo actualizar así la pantalla táctil del mismo.

```
  if(!mFirstConnection){
            String pressm = "a";
            mBluetoothConexion.send(pressm);
        }
```

4.2.5. Pantallas de los modos de operación

En términos de conexión, como ya se ha descrito con anterioridad, ésta debe de estar establecida para acceder a los distintos modos de operación y por ello no es necesario reconectar en cada actividad. Únicamente se hará una llamada a los métodos `bindService()` y `unBindService()` en cada uno de los modos de operación cuando corresponda:

- En el estado *onStart()* se llamará al método `bindService()`

```
Intent intent = new Intent(this, BluetoothConexion.class);
bindService(intent,mConnection, Context.BIND_AUTO_CREATE);
```

- En el estado *onStop()* se llamará al método `unBindService()`

```
unbindService(mConnection);
```

4.2.5.1. Pantalla del modo continuo

A la pantalla que se denomina modo continuo se accederá desde el botón con la imagen de las **tijeras** de la pantalla principal. Esta actividad está enfocada a manejar el cabezal de corte correspondiente existente en PRESSMATIC.

En este modo el usuario podrá seleccionar hasta tres velocidades (lenta, media o rápida) para llevar a cabo el cizallado y será capaz de accionar posteriormente el dispositivo PRESSMATIC pulsando el botón de START/STOP. La ejecución en este modo no se parará hasta que se vuelva a pulsar este último botón o se vuelva a la pantalla anterior. Cabe señalar, que la velocidad se puede cambiar en tiempo real, a la vez que realiza la operación de corte. Esto beneficiará la modificación de parámetros mientras se trabaja facilitando así el proceso, de tal forma que el usuario no tenga que movilizar el dispositivo PRESSMATIC hacia sí mismo, lo cual interrumpiría la operación.

Figura 31. Pantalla del modo continuo

Al igual que en la pantalla principal, cada uno de estos botones enviará un carácter al dispositivo PRESSMATIC de tal manera que se tenga constancia de las acciones del usuario. Como ejemplo, se muestra el listener del botón que asigna la velocidad lenta:

```
VelocLenta.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
            String lenta = "5";
            // EnvÍo del String que corresponda
            mBluetoothConexion.send(lenta);

            }
        });
```

En cuanto a la interfaz, se encontrarán elementos comunes en esta pantalla y la pantalla del modo de operación paso a paso, ya que son muy similares. La única peculiaridad de estas

dos pantallas es que los botones que asignan la velocidad se han establecido en la vista mediante un *RelativeLayout*. El fragmento de código XML que implementa estos tres botones, localizado en el archivo *Pressmatic/res/layout/modocontinuo.xml*, es el siguiente:

```
<RelativeLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="20dp">
                <Button
                android:id="@+id/BotonLenta"
                android:layout_width="85dp"
                android:layout_height="85dp"
                android:background="@drawable/speed_1_rojo"/>
                <Button
                android:id="@+id/BotonMedia"
                android:layout_width="85dp"
                android:layout_height="85dp"
                android:layout_toRightOf="@id/BotonLenta"
                android:background="@drawable/speed_2_rojo"
                android:layout_marginRight="15sp"
                android:layout_marginLeft="15sp"/>
                <Button
                android:id="@+id/BotonRapida"
                android:layout_width="85dp"
                android:layout_height="85dp"
                android:layout_toRightOf="@id/BotonMedia"
                android:background="@drawable/speed_3_rojo"/>
</RelativeLayout>
```

Se puede apreciar que después de declarar el primer botón los otros dos llevan asignados el atributo `layout_toRightOf` , que los dispondrá a la derecha del botón elegido.

Como se ha expuesto en el apartado de diseño de los iconos, se puede observar que el botón de START/STOP presenta una mayor anchura para facilitar al usuario la pulsación del mismo. En la implementación de las siguientes pantallas se encontrarán más botones que comparten esta característica.

4.2.5.2. Pantalla del modo paso a paso

Esta pantalla es la encargada de controlar tanto las **pinzas grandes** o de agarre como las **pinzas pequeñas** mediante los cabezales correspondientes y, por lo tanto, se accederá a ella mediante estos dos botones de la pantalla principal.

Como en el modo continuo, el usuario podrá escoger la velocidad deseada de la misma manera para llevar a cabo los movimientos de las pinzas. En cambio, en este caso se podrá realizar la apertura de la pinza mediante los botones de ABRIR y CERRAR que se disponen en la pantalla. Con estos botones se producirá una mayor o menor apertura de la pinza con

cada pulsación, dependiendo de la velocidad seleccionada, favoreciendo así un posicionamiento de las pinzas más preciso por parte del usuario.

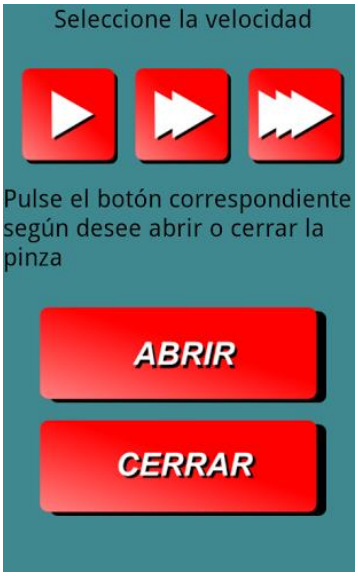

Figura 32. Pantalla del modo paso a paso

4.2.5.3. Pantalla del modo de corte único

Esta pantalla es la que presenta una mayor simplicidad en su implementación, ya que en este modo de operación se realiza un ciclo temporizado y no existe modificación posible de los parámetros de trabajo. Para acceder a este modo el usuario pulsará el icono del **cortaúñas** que le permitirá utilizar la herramienta en cuestión.

Figura 33. Pantalla del modo de corte único

4.2.6. Preferencias de usuario

Mediante esta pantalla, a la cual se accede mediante el botón correspondiente desde la actividad PRESSMATIC, el usuario es capaz de modificar una serie de parámetros que le permitirán un uso más personalizado de la aplicación. Estos parámetros son tales como el color de la interfaz y el tamaño de fuente. A continuación, se procede a realizar una descripción de las opciones que integra.

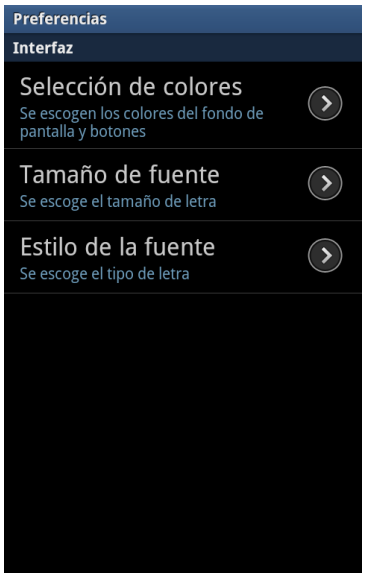

Figura 34. Preferencias de usuario

- **Letra**

Con el fin de adaptar de la manera más efectiva posible los criterios de accesibilidad existentes, se añade la oportunidad de alterar por parte del usuario tanto el tamaño como el estilo de la letra de los distintos textos de la aplicación. Aunque la cantidad de textos es reducida debido a que se ha querido realizar una aplicación lo más intuitiva posible, estas modificaciones se plantean como una opción.

Entre las ventajas que presenta el cambio del tamaño de letra destaca la posibilidad de aumentar el tamaño del texto para hacerlo más visible o de reducirlo para facilitar un uso más cómodo de la misma una vez que se sabe cómo utilizarla. Por otro lado, cambiar el estilo de la letra permitirá al usuario seleccionar aquel tipo de fuente con el que se encuentre más cómodo.

- **Color**

Para crear una versión compatible para todos los móviles Android se ha hecho necesario introducir colores personalizados debido a que hay temas no aplicables a todas las versiones. Estos colores han sido definidos en el entorno de desarrollo mediante una herramienta que se proporciona y permite, mediante una paleta de colores, seleccionar el color y el brillo que se desea para cada caso.

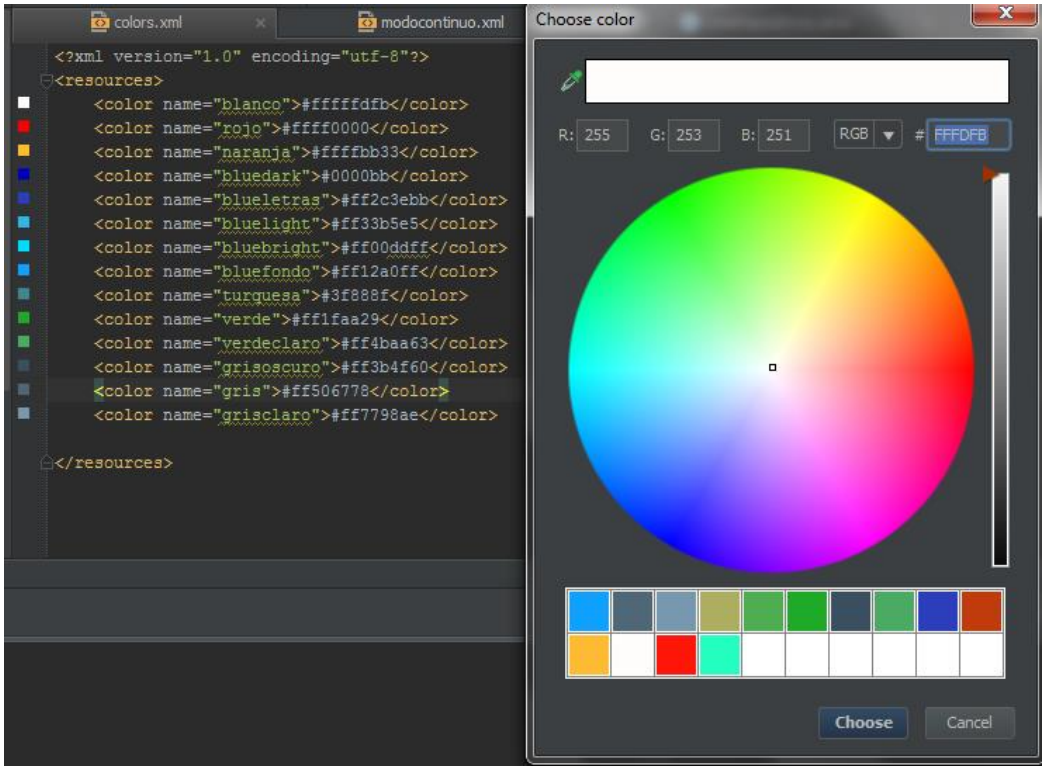

Figura 35. Selección de colores en IntelliJ IDEA

Si se desea asignar un color a un elemento en particular de una vista se debe concretar el atributo `android:color` o, en el caso de un texto ,`android:textColor`. Si se pretende utilizar un color existente en la biblioteca de Android, la asignación se realiza de manera distinta en comparación a si se quiere utilizar un color definido por el desarrollador, a continuación se muestra la diferencia:

`android:textColor="@android:color/black"` está definido por Android.
`android:textColor="@color/rojo"` se indica que el color en cuestión se encuentra definido en el archivo de recursos correspondiente, en este caso *res/values/colors.xml*.

Como ya se ha comunicado, existen dos sets de botones distintos, rojo y azul. Teniendo en cuenta esto se han puesto también a disposición del usuario hasta cuatro configuraciones de color para las distintas pantallas. Dos de estas configuraciones incluyen botones rojos, una con el fondo verde turquesa, que se muestra en imágenes anteriores, y otra con un fondo azul. Las otras dos configuraciones incluyen botones azules con fondos blanco grisáceo y naranja. La elección de estos colores ha sido motivada por una búsqueda de contraste, provisto el círculo cromático presentado anteriormente en la Figura 18.

- **Idioma**

Ya que se desea que la aplicación sea utilizada por el mayor número de usuarios posible, se facilita el cambio de idioma entre castellano e inglés, que modificará todos los textos y menús al idioma elegido.

Para que esto sea posible resulta imprescindible incluir, para cada idioma, los textos (*Strings*) de los diferentes elementos de la inferfaz en un archivo distinto. Estos archivos se localizan generalmente en la carpeta *NombreDeLaAplicacion/res/values/* pero, si se quieren añadir más idiomas, será necesario crear carpetas especializadas para cada uno.

Para permitir el uso de otro idioma se creará una carpeta denominada *values*, seguida del prefijo del idioma. Por ejemplo, para el caso del idioma inglés se creará la carpeta denominada *values-en*. En este archivo se creará el fichero de *Strings* en inglés, que modificará todos los textos de la aplicación únicamente cuando el teléfono cambie de idioma. Por si existieran dudas, la manera de cambiar el idioma de un dispositivo Android implica generalmente seguir la siguiente ruta: *Ajustes del teléfono>Teclado e idioma>Ajustes de idioma>Seleccionar idioma*.

Algunas de las modificaciones que se pueden apreciar en las actividades tras cambiar los parámetros disponibles son visibles en las figuras mostradas a continuación:

Figura 36. Ejemplo de cambio de parámetros 1 y 2

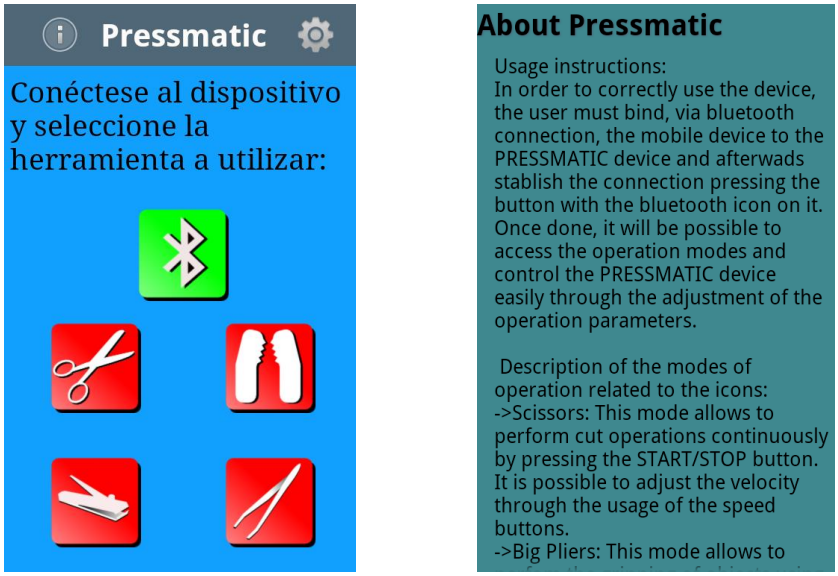

Figura 37. Ejemplo de cambio de parámetros 3 y 4

Se observará que al modificar los parámetros se produce un reajuste de los elementos de la pantalla gracias al ya conocido atributo `layout_weight`.

4.2.6.1. Lectura de las preferencias de usuario

Para hacer una lectura de las preferencias se crea una clase especialmente destinada a cumplir este objetivo denominada LecturaPref. Esta clase contiene una serie de métodos que permiten reestablecer los elementos de cada una de las vistas de usuario según la configuración elegida y actualizar los tamaños de los textos de las mismas. En esta clase cada uno de los métodos existentes obtendrá las preferencias de usuario mediante el método `getDefaultSharedPreferences()` y actuará en consecuencia, según corresponda.

El primero de estos métodos determina los elementos de la interfaz para cada una de las actividades. Esto es necesario debido a que en cada actividad el método `setContentView()` establece un Layout diferente, una vista distinta. Si en las preferencias de usuario se pueden encontrar hasta cuatro configuraciones de iconos y colores, para el modo continuo, por ejemplo, tienen que existir cuatro archivos layout que las definen. Para este modo en particular, el método para actualizar la interfaz de ese modo es el siguiente:

```
public static void defineInterfazContinuo(Activity activ){

        SharedPreferences  prefs =
        PreferenceManager.getDefaultSharedPreferences(activ);
        // Interfaz por defecto 2
        String interfazElegida = prefs.getString("color", "2");
        // Se leen las preferencias y se establece el layout elegido
        if (interfazElegida.equals("1")){
            activ.setContentView(R.layout.modocontinuo);
        } else if (interfazElegida.equals("2")){
            activ.setContentView(R.layout.modocontinuo2);
        }else if (interfazElegida.equals("3")){
            activ.setContentView(R.layout.modocontinuo3);
        }else if (interfazElegida.equals("4")){
            activ.setContentView(R.layout.modocontinuo4);
        }
    }
```

Se podrán hallar también el resto métodos utilizados para asignar la interfaz de las demás actividades en el archivo de la clase LecturaPref.java, que son muy similares al mostrado.

Por otra parte, para actualizar el tamaño de los distintos textos (`TextView`) de la aplicación que se desee, se utilizará otro método. Según el tamaño elegido entre las opciones existentes se establecerá el tamaño de letra del texto que se desee modificar. Los tamaños de letra serán, como se aprecia en el siguiente fragmento de código, 18sp, 22sp o 26sp como máximo. Estos tamaños se corresponden con los tamaños pequeño, mediano y grande que seleccionará el usuario en la aplicación. El código de dicho método se adjunta justo debajo:

```
public static void actualizaTamañoLetra(TextView textView, Context
context){

        SharedPreferences prefs =
        PreferenceManager.getDefaultSharedPreferences(context);
        float valorTamañoLetra;
        // Tamaño de letra por defecto: selección 2
        String tamañoLetra = prefs.getString("tamaño letra", "2");
        // Lectura de preferencias y selección del tamaño
        if (tamañoLetra.equals("1")){
            valorTamañoLetra = 18;
        } else if (tamañoLetra.equals("2")){
            valorTamañoLetra = 22;
        } else {
            valorTamañoLetra = 26;
        }
        // Se establece el tamaño elegido
        textView.setTextSize(valorTamañoLetra);
    }
```

El último método para obtener las preferencias de usuario es el encargado de actualizar la fuente de la letra según la elección del usuario. Este método es idéntico al que actualiza el

tamaño de letra, solo que en este caso se modifica el tipo de letra, variando entre SANS_SERIF, SERIF y MONOSPACE, como se puede apreciar a continuación:

```
public static void actualizaFuenteLetra(TextView textView, Context
context){
        SharedPreferences prefs =
        PreferenceManager.getDefaultSharedPreferences(context);
        // Fuente de letra por defecto: selección 1
        String fuenteLetra = prefs.getString("fuente letra", "1");
        // Se leen las preferencias y se establece el estilo elegido
        if (fuenteLetra.equals("1")){
            textView.setTypeface(Typeface.SANS_SERIF);
        } else if (fuenteLetra.equals("2")){
            textView.setTypeface(Typeface.SERIF);
        } else {
            textView.setTypeface(Typeface.MONOSPACE);
        }

    }
```

4.2.7. Pantalla de ayuda/información

En la aplicación se incluye una pantalla de ayuda en la que se muestra información genérica acerca de la aplicación Android. Entre estos datos se puede encontrar a quién va dirigida, su fin de utilización y una descripción de cómo usar la aplicación de manera adecuada y satisfactoria. En particular, se incluyen instrucciones de uso de los distintos modos de operación implementados a los que accede el usuario mediante los correspondientes iconos.

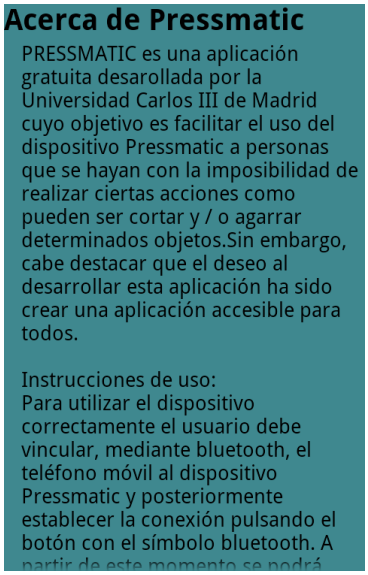

Figura 38. Pantalla de ayuda/información

4.2.8. Pantallas apaisadas

En el transcurso del proceso de diseño e implementación, se ha intentado que todas y cada una de las pantallas de la aplicación se asemejen a las existentes en la pantalla táctil del dispositivo PRESSMATIC. Este proceso de coordinación de pantallas, como se ha

comentado con anterioridad, se ha realizado con la colaboración de Rodrigo Martín, encargado de la programación de dicha pantalla táctil.

Después llevar a cabo algunas pruebas con las interfaces de usuario de los dos dispositivos, en seguida sería fácil percatarse que serían las pantallas apaisadas o pantallas horizontales de la aplicación Android las que, por parecido a la pantalla táctil, presentaran más similitudes con la misma. Esto es así ya que la forma en la que se opera con la pantalla táctil de PRESSMATIC se asemeja más a una pantalla dispuesta de esta forma, como se aprecia en la figura 39.

Figura 39. Modelo de dispositivo PRESSMATIC

Cabe destacar que las versiones apaisadas de las pantallas de la aplicación se generan de manera automática al implementar las pantallas habituales, las pantallas verticales. Sin embargo, en la mayoría de ocasiones, los elementos se encuentran dispuestos de una manera incorrecta, los textos no se ven correctamente o surgen otro tipo de inconvenientes.

Conocido esto, se hace indispensable definir de nuevo las actividades en las que existen problemas de este tipo, dando lugar a una nueva versión de las pantallas, que se creará en la carpeta *Pressmatic/res/layout-land*. Para el caso de la aplicación ha sido necesario crear estas versiones para cuatro actividades: la pantalla principal de la aplicación, la pantalla del modo continuo, la pantalla del modo paso a paso y la pantalla de presentación. En el resto de pantallas la visualización de los elementos es correcta y se pueden utilizar sin problema.

En la siguiente hoja se podrán observar, al lado izquierdo, dos de las pantallas que el usuario podrá contemplar cuando use el dispositivo PRESSMATIC y, al lado derecho, las dos pantallas apaisadas de la aplicación Android que se corresponden con estas.

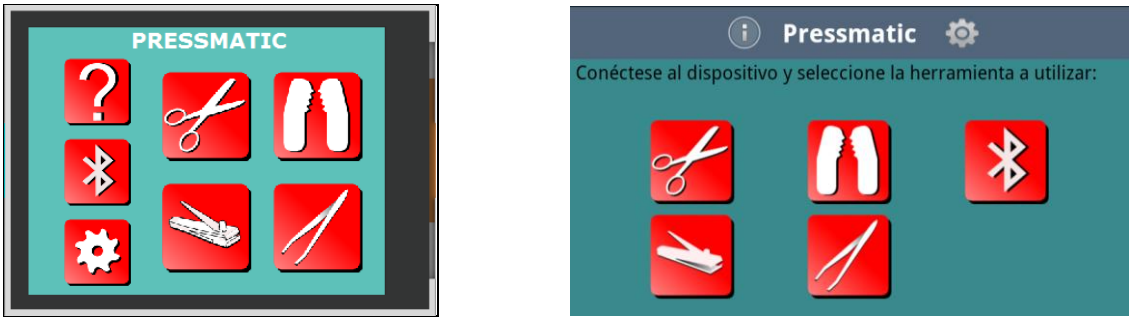

Figura 40. Pantalla táctil de Pressmatic 1 [4] y pantalla apaisada de la aplicación 1

Figura 41. Pantalla táctil de Pressmatic 2 [4] y pantalla apaisada de la aplicación 2

Capítulo 5

Evolución de la aplicación

Este capítulo está destinado a detallar los cambios y pruebas que se han ido realizando desde que se inició el proyecto, y a explicar las decisiones de diseño tomadas que han llevado a la aplicación PRESSMATIC al estado de desarrollo actual en el que se encuentra.

5.1. Interfaces experimentales. Ideas desechadas

A lo largo del proceso de diseño se han ido realizando modificaciones según se han ido perfilando las necesidades de los usuarios y los criterios finales de la aplicación. En esta sección se detallarán los principales cambios que se han producido en varias facetas de la aplicación desde su primer diseño.

5.1.1. Navegación entre pantallas

Inicialmente en la aplicación se implantó que el usuario realizara una selección del modo de operación, independientemente de la herramienta a utilizar, como se muestra en la figura 42. Sin embargo, este diseño se desechó rápidamente, ya que se advirtió que la elección del modo de operación, como son el modo continuo y el modo paso a paso, podría producir confusión y dificultad a la hora de utilizar el dispositivo para un usuario no familiarizado con dichos conceptos.

Figura 42. Primera versión pantalla principal y pantalla de selección de modo

Observando estas figuras y comparándolas con la aplicación final es posible percatarse de que ha habido una notable evolución de la aplicación. En versiones anteriores de PRESSMATIC el usuario necesitaba atravesar hasta tres pantallas para lograr encontrarse en el modo de operación deseado. Esto es debido a que en un principio PRESSMATIC no estaba tan centrada en el usuario como debería. El motivo es que, ante la falta de experiencia en el desarrollo de aplicaciones, se creó una aplicación más centrada en el desarrollador de la misma que en los destinatarios, originando así que la navegación entre las distintas pantallas se tratara de algo confuso y poco intuitivo para aquellos poco familiarizados con su funcionamiento.

En relación a esto, una de las mayores mejoras en la accesibilidad de la aplicación se ha debido a la introducción de **botones con iconos**. Estos facilitarían al usuario escoger el modo de funcionamiento mediante la elección de la herramienta. De esta forma se realizaría una **selección intuitiva** aún sin haber utilizado nunca el dispositivo PRESSMATIC. Gracias a este avance se pudo **reducir** el número de **pantallas** por las que el usuario tenía que pasar, sustituyendo las dos actividades mostradas previamente por una sola, añadiendo así otra gran ventaja.

Asimismo, se puede apreciar también, tanto en las figuras mostradas antes como en figuras que se verán más adelante, una mayor cantidad de texto e instrucciones en la interfaz de usuario. Sabido que se desea crear una aplicación lo más intuitiva posible, se persigue el diseño de una aplicación que el usuario sepa utilizar sin necesidad de instrucciones, carente de texto o cuyo texto sea mínimo. Es por esto por lo que resulta esencial la inclusión en la aplicación de una pantalla de información y ayuda, a la que el usuario acudirá en primera instancia pero, más tarde, en posteriores ejecuciones de la aplicación, no le será necesario.

5.1.2. Menú

La aplicación PRESSMATIC contenía inicialmente un menú al que se podía acceder mediante la pulsación del botón correspondiente en cada dispositivo. En dicho menú se mostraban al usuario una serie de opciones, que eran las siguientes: “Conectarse a un dispositivo”, “Acerca de…” y “Preferencias”, como se aprecia en la figura 43. En las versiones iniciales la pantalla principal de PRESSMATIC no constaba de un botón que permitiera conectar a PRESSMATIC y desconectar siempre que se deseara, ya que este rasgo se presentaba de manera única en el menú.

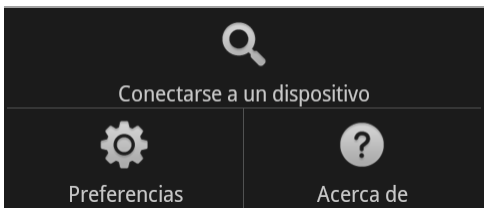

Figura 43. Menú existente en versiones antiguas

Más tarde dicho menú sería desechado por un simple motivo: los botones incluidos en él podían ser incluidos en la interfaz de PRESSMATIC sin la necesidad de su existencia.

Con la adición del botón bluetooth y dado que los botones de preferencias e información estaban ya localizados en la parte superior de la pantalla, se procedió a la eliminación del menú. Así se favoreció la accesibilidad de la aplicación, ya que si antes había que pulsar el botón correspondiente del dispositivo Android con el fin de hacer el menú visible y establecer la conexión, esto dejaría de ser necesario. De tal forma se ahorró al usuario un paso más a la hora de utilizar la aplicación.

Cabe mencionar que desde la página de desarrolladores de Android se recomienda la utilización de la denominada *ActionBar* (figura 44), un elemento de diseño similar al utilizado, por varios motivos. La *ActionBar* [18] provee al usuario de un espacio definido para dar a la aplicación una identidad e indicar al usuario el estado en el que se encuentra la misma, facilita el acceso a acciones de una manera predictiva y ayuda a navegar por la aplicación. Es por estos motivos por el que se ha optado por incluir en la aplicación una característica similar a la *ActionBar* mediante una barra, en este caso creada por el desarrollador, que contiene los botones oportunos.

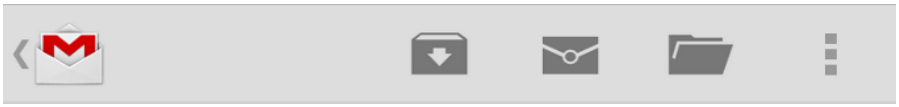

Figura 44. Ejemplo de ActionBar

5.1.3. Selección de parámetros

Algunas de las pantallas que más se han conseguido simplificar han sido las pantallas de los modos de operación. Aparte de hacer su selección algo mucho más sutil por parte del usuario con los iconos de la pantalla principal, se han hecho mejoras a la hora de escoger los parámetros de trabajo. Como se puede ver en el caso del modo continuo (figura 45), inicialmente se implementó la denominada barra de progreso o "*Seekbar*" posibilitando al usuario escoger entre varias velocidades:

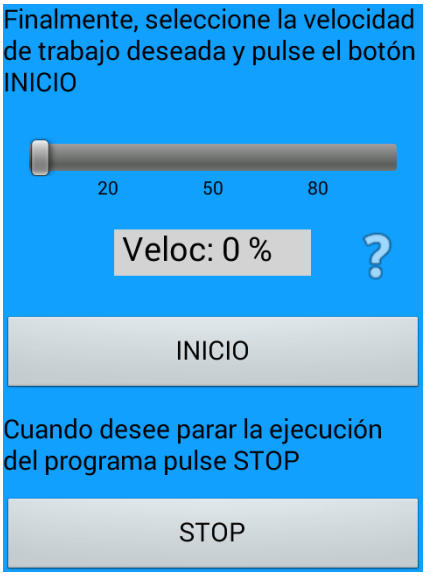

Figura 45. Selección de parámetros en versiones antiguas

A pesar de no ser una mala opción, se consideró una opción mucho más sencilla presentar tres botones que seleccionaran la velocidad, como en el diseño final. Los botones, al contrario que la barra, pueden ser cómodamente pulsados por todos los usuarios y, además, favorecen una selección más intuitiva de la velocidad. Con ellos los usuarios no se tendrían que preocupar, por ejemplo, de si la velocidad máxima es demasiado alta o si, por otro lado, a velocidades bajas las tareas tienen riesgo de no desarrollarse correctamente.

Asimismo, se puede apreciar que existen dos botones, uno de INICIO y otro de STOP, cuando en el diseño final solo existe un único botón que desempeña la misma función.

5.2. Pruebas de funcionamiento

Las pruebas de funcionamiento llevadas a cabo han ido enfocadas principalmente a conseguir que el protocolo bluetooth cumpla una serie de requisitos (apdo 1.3). Esto no significa que no se hayan efectuado pruebas en la interfaz de usuario, sino que la conexión bluetooth es un proceso más sofisticado y requirió un desarrollo en grupo.

5.2.1. Bluetooth

Como ya se comentó en el apartado de la conexión bluetooth (apdo 0), se barajaron tres posibilidades para implementarlo que dieron lugar a distintos ensayos. Los ensayos de conectividad bluetooth realizados se hicieron con una placa Arduino UNO al que se conectaba el módulo Bluetooth HC-05 de la manera que muestra el esquemático siguiente:

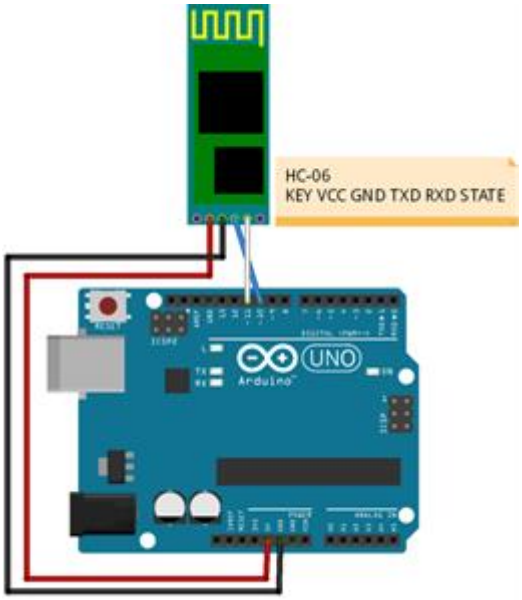

Figura 46. Esquemático de conexión [3]

Cuando el módulo esté conectado, el próximo paso consiste en establecer un puente entre la salida serial de dicho módulo y el monitor serial del entorno Arduino, que facultará la visión de los resultados en la pantalla del ordenador. Para proceder con los ensayos se debe cargar

al Arduino el código BT_Reciever_Monitor [3], que permite desempeñar el proceso descrito.

En el mencionado apartado se habla de manera general de las opciones testeadas. Se planteó como una alternativa inicial bastante sólida la posibilidad de implementar la aplicación en una sólida actividad, pero pronto se percibió que esto no era factible. Aunque la comunicación se realizaba con efectividad, las necesidades del usuario no se verían satisfechas de esta manera, haciéndose necesario el empleo de más actividades.

Por otro lado, se idea como una posible opción que la conexión se reestableciera con cada cambio de pantalla, ya que con el protocolo actual no se podía instaurar una conexión estable entre actividades. Esta alternativa permitía iniciar la conexión pero de un modo excesivamente ineficiente, ya que se requería esperar unos 10 segundos al cambiar de actividad para volver a conectar. Después de este periodo de tiempo se podría volver a enviar los comandos necesarios.

Sería por tanto la última de las alternativas contempladas, destinada a chequear la efectividad de la conexión basada en el Service, la que ofreciera mayores expectativas para cumplir los objetivos fijados. Una vez conocidos el funcionamiento y la implementación de los servicios en Android se obtuvieron resultados positivos rápidamente, que llevarían a escoger este método como forma de conexión. De esta manera se pudo conseguir una conexión estable entre actividades.

5.2.2. Interfaz

El tiempo dedicado a la realización de pruebas de la interfaz de usuario ha sido un proceso constante. Como se detalla más arriba, la interfaz ha ido evolucionando hasta cumplir los objetivos que se requerían de la mejor manera posible.

La manera de proceder ha consistido principalmente en ir agregando elementos factibles de mejorar la interfaz, en detrimento de otros. Posterior a esta incorporación de elementos se efectuarían los ensayos oportunos y se determinaría si la interfaz de usuario se ha visto mejorada o no con los cambios.

Este proceso de “evolución constante” de la aplicación es el que se ha seguido hasta alcanzar el diseño que se ha considerado como aquel que satisface las necesidades de usuario de una manera más efectiva.

Se puede destacar el botón bluetooth como un elemento que ha sido testeado en profundidad y que finalmente ha supuesto una mejora de la interfaz. Este botón ha presentado serios problemas en varias situaciones, como pueden ser almacenar el estado en el que se encuentra el mismo al modificar la interfaz o cuando se cambiaba el móvil a estado apaisado. Este hecho puede parecer poco relevante, pero en términos del protocolo de conexión llegó producir serios fallos. Uno de los fallos que se solventaron fue que, al tumbar el móvil una vez conectado, se perdía el estado del botón volviendo al estado inicial

y, en consecuencia, la conexión se veía destruida. Causa de esto era que el botón al volver a dicho estado eliminaba toda conexión, procediendo tal y como si el usuario hubiera pulsado el mismo para desconectar. Se descubrió que esto es debido a que toda actividad, al cambiar a modo apaisado, vuelve a crear la actividad de cero como si fuera una nueva actividad, pasando así por el método *onDestroy()* y después por el *onCreate()* otra vez.

Por otro lado, se han llevado a cabo una serie de pruebas con el fin de hacer compatible la aplicación con una amplia variedad de dispositivos, independientemente del tamaño de la pantalla o de la posición de los mismos.

Como ya se ha comentado previamente, para que los elementos de la interfaz aparezcan correctamente en todos los dispositivos, se hace necesario determinar en cada pantalla y para cada elemento el peso justo que va a ocupar cada uno sobre el total. Este reajuste de elementos, según los distintos tamaños de las pantallas existentes en Smartphone y Tablet, se ha realizado empleando el simulador de IntelliJ IDEA. Esta tarea ha requerido más tiempo del que inicialmente se esperaba, pero se han conseguido finalmente resultados satisfactorios, visibles en figuras anteriores.

5.3. Diseño final y motivos de elección

La elección del diseño final ha sido motivada por un trabajo constante tanto en el diseño de la aplicación, como en la comunicación bluetooth. Esta tarea se ha realizado siempre buscando una correcta adaptación de los criterios de accesibilidad existentes aplicables a la tecnología móvil y, en particular, a la plataforma Android.

Cabe mencionar que el creador de la tesis en la que se basa este proyecto [1], Gabriel María, se ha volcado para hacer de la aplicación PRESSMATIC una aplicación accesible, especialmente para las personas que han perdido gracilidad manual. Con su ayuda se han podido probar las distintas mejoras y facetas que se han ido incluyendo en la aplicación, ya que se trata de un usuario potencial de la misma. En parte gracias a esto se ha podido construir la aplicación que aquí se presenta de una forma más completa y satisfactoria.

Capítulo 6

Presupuesto

En este apartado se describen de manera detallada los recursos necesarios para ejecutar el proyecto y los costes asociados a los mismos.

6.1. Presupuesto

El presupuesto necesario para desarrollar el proyecto se dividirá en varios conceptos como son los costes de personal, costes de equipo, costes de material y otros costes.

Para determinar con precisión cada uno de estos costes, en primer lugar es indispensable conocer en qué consisten cada uno de ellos.

- Costes directos:
    - Costes personales: son los costes asociados a las personas que trabajan en el desarrollo del proyecto, siendo necesario detallar la cantidad de horas de trabajo dedicadas.
    - Herramientas hardware: costes proporcionales al tiempo de uso de las distintas herramientas hardware utilizadas como pueden ser el ordenador necesario para llevar a cabo la programación o el dispositivo Android para ejecutar las pruebas (amortización de hardware).
    - Herramientas software: costes proporcionales al tiempo de ejecución del proyecto en términos de licencias software requeridas (amortización de software).
    - Materiales fungibles: costes relativos al material de oficina utilizado o los recursos de impresión necesarios.
- Costes indirectos
    - Costes relativos a factores como el alquiler de inmuebles, dirección del proyecto, coste en suministros, etc. Se ha estimado su valor en un 20% del total de los costes directos atribuidos al proyecto.

Ahora se especificarán cada uno de los costes asociados al proyecto, descritos anteriormente:

- Para el desarrollo del proyecto se ha requerido un graduado en Ingeniería de Tecnologías Industriales dedicado al mismo un tiempo aproximado de 6 meses. Si por cada uno de estos meses se contabilizan 20 días laborales, el número de días resultantes asciende a 120 días. Para calcular el total de horas trabajado se tendrá en cuenta una jornada laboral de 5 horas, lo que significa un total de 600 horas de trabajo. Estos datos junto con el valor de la medida hombre mes (131.25 horas) dan lugar a los resultados de la tabla 3:

Costes de personal				
Nombre	Categoría	Dedicación (hombre mes)	Coste hombre mes (€)	Coste total (€)
Alejandro Vega Gómez	Graduado en Ingeniería de Tecnologías Industriales	4,57	2400	**10971,43**

Tabla 3. Costes de personal

- Los costes relativos a las herramientas hardware utilizadas se detallan en la tabla 4:

Costes hardware					
Descripción	Coste (€)	% Uso dedicado	Dedicación (meses)	Periodo de depreciación	Coste imputable* (€)
Ordenador portátil	800	75	6	60	60
Samsung Galaxy S Advance	300	75	6	48	28,12
Módulo bluetooth HC - 05	4,5	100	4	24	0,75
Arduino UNO Rev3	20	60	4	60	0,8

Total: 89,67 €

Tabla 4. Costes hardware

* Fórmula de amortización:

$$Coste\ de\ amortización = \frac{A}{B} x\ C\ x\ D$$

Dónde: $A=$ número de meses desde la fecha de facturación en que el equipo es utilizado
$B=$ periodo de depreciación (dependiente del dispositivo, en general, 60 meses)
$C=$ coste del equipo (sin IVA)
$D=$ % de uso dedicado de ese equipo al proyecto

- Los costes relacionados con el software empleado se detallan en la tabla 5:

Costes software				
Descripción	Coste licencia (€)	Duración de la licencia	Tiempo de uso en el proyecto (meses)	Coste imputable* (€)
S.O. Windows 7 Home and Student	90	Indefinida	6	9
Paquete Office 2010 Home and Student	120	Indefinida	4	18
IntelliJ IDEA Community Edition 13.0.2	0	Indefinida	6	0
Plataforma Android 2.3.3 (SDK)	0	Indefinida	6	0
Inkscape	0	Indefinida	4	0
				Total: 27€

Tabla 5. Costes software

*En este caso para calcular el coste asociado a las licencias, dado que su duración es ilimitada, se ha considerado el tiempo que tarda un software como el utilizado en quedarse obsoleto. Se han estimado 5 años para hacer este cálculo, dando lugar a un porcentaje de coste imputable de un 10% para el caso (sobre 6 meses) para el caso del sistema operativo y un 15% (sobre 4 meses) para el caso del paquete Office.

-Los costes asociados a materiales fungibles como puede ser material de impresión, material de papelería, encuadernación… se han valorado en **50 €**.

Una vez conocidos los costes directos asociados al desarrollo del proyecto se pueden calcular de manera automática los costes totales asociados a éste, ya que los costes indirectos se calculan como un porcentaje de los costes directos.

La tabla 6 contiene los costes totales relativos al desarrollo del presente proyecto:

Costes totales	
Concepto	Presupuesto (€)
Costes de personal	10971,43
Costes de amortización (Hardware y Software)	116,675
Costes fungibles	50
Costes indirectos	2227,62
Total	**13365,72**

Tabla 6. Costes totales de desarrollo

Capítulo 7

Conclusiones y líneas futuras

En este capítulo se muestran las conclusiones resultantes del desarrollo del presente proyecto, tomando como base los objetivos que se han planteado al comienzo del mismo, y analizando el resultado final obtenido después del proceso de desarrollo.

Para finalizar, se expondrán las posibles mejoras y trabajos futuros que se propone desempeñar para este proyecto.

7.1. Conclusiones del proyecto

El objetivo principal de este proyecto, el desarrollo de una aplicación Android accesible que posibilite llevar a cabo el control práctico del dispositivo PRESSMATIC, se ha cumplido de una manera satisfactoria. No obstante, se hace necesario evaluar el grado de cumplimiento de los principales objetivos:

- Control simple e intuitivo: Un requisito imprescindible a la hora de desarrollar cualquier tipo de software es facilitar al usuario la utilización del mismo, de tal manera que sus necesidades se vean cumplidas de una manera directa y efectiva.

Esta condición se cumple en gran medida gracias al trabajo coordinado de equipo que se ha profesado a la hora de desarrollar tanto el protocolo de comunicación como de perfeccionar la interfaz.

Como ya se ha descrito, la implementación del objeto *Service* de la API de Android ha posibilitado la comunicación con PRESSMATIC de la forma más sencilla en términos del usuario. A fin de cuentas, el usuario sólo deberá establecer la conexión en un principio y a partir de ahí podrá centrarse en la tarea que desee ejecutar.

Por otro lado, la mejora de la interfaz ha llevado, entre otras cosas, a reducir el número de pasos que el usuario debe realizar para controlar PRESSMATIC. Asimismo, la sintonía existente entre la pantalla del dispositivo Android y la del dispositivo PRESSMATIC genera una navegación mucho más clara para el usuario. Al usuario le bastará entonces con saber manejar uno de los dos dispositivos para, de manera automática, saber controlar el otro.

Véase que, si un usuario manejara solo la aplicación sin tener noción del dispositivo PRESSMATIC ni de sus funcionalidades, puede que los iconos no resultaran

suficientemente claros. Sin embargo, la aplicación está enfocada a los usuarios de PRESSMATIC, que se supone conocen la funcionalidad del dispositivo y, si todavía surgieran dudas entre éstos, la aplicación contiene una pequeña guía de utilización en la pantalla de ayuda en el caso fuera precisa.

- Accesible para todos: Hacer de la accesibilidad un rasgo de la aplicación ha sido uno de los principales objetivos. Cabe destacar que desde el principio se ha intentado que la aplicación sea accesible, no sólo para aquellos usuarios que debido a su pérdida de movilidad precisan beneficiarse de las funcionalidades de PRESSMATIC directamente, sino para cualquier persona.

En la aplicación se han intentado incluir la mayor cantidad de atributos de accesibilidad a disposición del desarrollador. Con esta meta se ha introducido la posibilidad de **personalizar** la interfaz de usuario, ya sea modificando los colores que la forman o cambiando la aparición de los textos en cuanto a tamaño y estilo. Se han añadido también recursos para sacar provecho de la función Talkback que Android proporciona en sus dispositivos y mediante el cual se facilita la navegación. A su vez, se ha añadido también la posibilidad de cambiar el idioma para que la aplicación sea utilizable por usuarios que desconozcan el idioma español.

Otra de las facetas de accesibilidad incluidas es que la **navegación** por la aplicación es **guiada**, es decir, el usuario no puede acceder a las pantallas de los modos de operación si la conexión no ha sido establecida previamente. De esta manera se consigue que el uso de la aplicación sea eficiente y no se incurra en fallos de utilización. Además, existe un sistema de **notificaciones** e **instrucciones** de usuario consistente, centrado principalmente en la conexión y destinado a proporcionar información al mismo cumpliendo con las pautas de accesibilidad.

Aún con todo esto, cabe mencionar que la cantidad de formas existentes de integrar la accesibilidad en una aplicación son muy diversas y no ha sido posible hacer uso de todas. Es por ello por lo que, pese a que la implantación de la accesibilidad en la aplicación ha sido un objetivo siempre presente, es difícil evaluar si la aplicación PRESSMATIC es plenamente accesible. No obstante, sí se puede afirmar con total seguridad que la aplicación está diseñada especialmente para aquellas personas que han sufrido una pérdida de movilidad manual y que otros usuarios también encontrarían grandes facilidades al hacer uso de ella.

7.2. Conclusiones personales

Desde que se me planteó este proyecto quise ponerme a prueba a mí mismo.

Siempre me ha llamado mucho la atención la cantidad de personas que disponen de un teléfono móvil y hacen uso del mismo y de sus aplicaciones de manera constante, sea donde sea. Es por esto por lo que me suscitaba mucha curiosidad saber si yo sería capaz de desarrollar una aplicación para Android. Sin embargo, mis conocimientos de esta plataforma eran nulos y no estaba familiarizado con la programación en Java.

Fue la propuesta de mi tutor la que me llevó a plantearme este reto y no se me ocurrió mejor manera de intentarlo que aceptando un proyecto que estuviera enfocado a ayudar a personas que encuentran dificultades a la hora de realizar ciertas acciones por sí mismas. Me he percatado de que realizar este tipo de proyectos aportan mucho personalmente y que, cuanto más te implicas y te esfuerzas, más recibes a cambio.

A lo largo de la realización del proyecto he aprendido muchísimo y, aunque haya sido un largo camino, ha merecido la pena. He vivido momentos desesperantes cuando desconocía como arreglar fallos o errores que surgían, pero son estos obstáculos con los que realmente se aprende y los que más satisfacción producen a posteriori. Otra de las cosas que he aprendido, tanto en este proyecto como a lo largo de la carrera, es que el trabajo en equipo no siempre es sencillo y hace falta coordinación para que las propuestas salgan bien.

En conclusión, se puede decir que con este proyecto he conseguido cumplir los objetivos que me propuse en un principio sin saber verdaderamente si iba a ser capaz de ello y es que, en definitiva, con esfuerzo todo se consigue.

Por último, me gustaría añadir que tanto la realización de este proyecto como los años de carrera han resultado una experiencia muy aportadora y han motivado que mi interés por la ingeniería crezca todavía más.

7.3. Trabajos y mejoras futuros

Esta sección expone algunas de las posibles mejoras y trabajos futuros que se podrían llegar a realizar en la línea de este proyecto de fin de grado:

- Como ya se planteaba al final del apartado 2.3.2, **crear una aplicación para iOs o Windows Phone** sería una manera idónea de abarcar una mayor cantidad de usuarios potenciales de la aplicación PRESSMATIC. Para cumplir el ambicioso objetivo de llegar a la mayor cantidad de usuarios posible, bastaría con desarrollar una aplicación con las mismas o más funcionalidades que la aplicación que contempla este proyecto.

- Implementar que la **conexión** se realice de manera **automática al dispositivo PRESSMATIC**, ahorrándole así al usuario un paso, favoreciendo un uso más directo de la

aplicación. Esta opción no ha sido contemplada, ya que se ha priorizado la mejora de la accesibilidad en la aplicación y la estabilidad en la comunicación bluetooth. Además, el hecho de establecer una conexión automática implica conectarse siempre al mismo módulo bluetooth, en este caso al módulo HC-05. De esta manera no se podrían efectuar pruebas con otros módulos y esto es recomendable únicamente cuando la aplicación PRESSMATIC sea definitiva.

Para llevar a cabo esta tarea se recomendaría crear una opción adicional en las preferencias de usuario, de tal manera que se escogiera entre establecer una "Conexión automática con el último dispositivo" o no establecerla. Esta opción, al estar seleccionada, estaría encargada de almacenar los datos de conexión del último dispositivo utilizado y produciría una conexión automática al mismo al pulsar el botón bluetooth. En caso de no estar seleccionada se mostraría la lista de dispositivos, como ocurre ahora.

- Se ha prestado especial atención al establecer la cantidad de espacio que ocupan los distintos elementos y la disposición de los mismos en la interfaz de usuario. Aun así, cabe mencionar que en los dispositivos Tablet, aunque PRESSMATIC funciona perfectamente, los botones no son proporcionales al tamaño de las pantallas de estos dispositivos. Esto significa que el tamaño de los botones de la aplicación que se podrán encontrar en un Smartphone y una Tablet es el mismo. Por este motivo sería altamente recomendable **implementar iconos personalizados para dispositivos Tablet** que estuvieran en concordancia con las pantallas de los mismos.

- La comunicación que se establece con PRESSMATIC se produce en un solo sentido, es decir, de la aplicación móvil con el dispositivo PRESSMATIC, cuando lo ideal sería una comunicación de doble sentido. Esto implicaría implantar un **modo escucha** también en la **aplicación Android**, de tal manera que las acciones producidas en la pantalla táctil del dispositivo PRESSMATIC produjeran el mismo efecto en la pantalla de la aplicación Android.

Cabe mencionar que la conexión actual conexión permite el uso de dicho modo ya que existe una pila de lectura de datos denominada InputSteam, pero no se ha llegado a implantar su funcionamiento.

- Si se desea presentar una forma alternativa al control de PRESSMATIC con la aplicación se podría implementar el **manejo de la aplicación** con el empleo de **comandos de voz**. Esto se podría realizar sacando provecho de la API de reconocimiento de voz de Google.

- Por supuesto, **dar a probar** la aplicación sobre PRESSMATIC a un **grupo** lo más **numeroso** posible de **usuarios** con el fin de escuchar sus sugerencias e ideas y mejorar así la usabilidad de la aplicación.

Anexo

Esta parte del proyecto está destinada a explicar de una manera detallada algunos de los procesos básicos relevantes al desarrollo que no se han incluido anteriormente, ya que no se consideraban de importancia primaria a la hora de detallar este proceso.

Anexo A. Instalación de IntelliJ IDEA

Antes de llevar a cabo la instalación del entorno de desarrollo es necesario cumplir una serie de requisitos iniciales que permitirán hacer uso del mismo.

En primer lugar, es imprescindible tener instalada la **máquina virtual de Java**, que permitirá hacer uso de dicha máquina en el ordenador. Esto se puede realizar desde la página de Java. Después de esto, es altamente recomendable instalar también el **JDK,** que permitirá hacer uso de cualquier software de desarrollo y utilizar herramientas adicionales. Para descargarlo se debe acceder a la página web de Oracle.

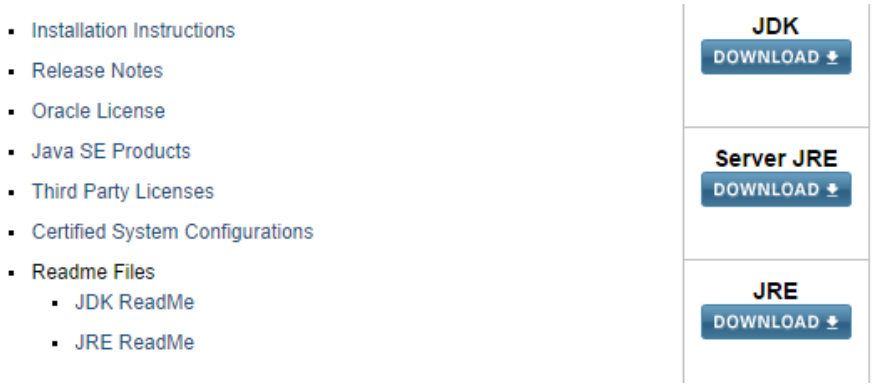

Figura 47. Descarga del JDK

Es importante que, tras proceder como aparece en la imagen, se descargue el JDK que se corresponda con el sistema operativo que se esté utilizando. En el caso de que no fuera así, podrían surgir problemas más adelante.

Como último paso antes de la instalación del entorno de desarrollo instalaremos **Android SDK de Google**. Se trata de un paquete proporcionado por los desarrolladores de Android y que incluye los elementos precisos para instalar de manera fácil el IDE. Esto incluye las plataformas de desarrollo y máquinas virtuales necesarias para las distintas versiones de Android. Posterior a la instalación se podrá ejecutar el programa SDK Manager y seleccionar los distintos paquetes a instalar, según se necesite.

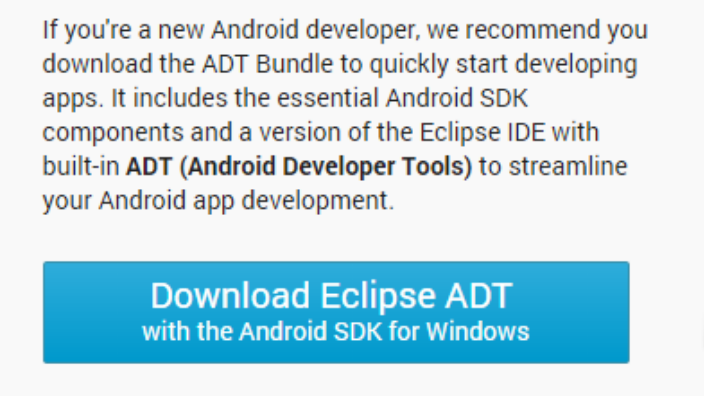

Figura 48. Descarga de Android SDK

Una vez se haya terminado la instalación de estos elementos se procederá a la instalación de IntelliJ IDEA. Para instalar IntelliJ IDEA se debe efectuar su descarga desde su página web oficial. Cuando se entre en la página, se puede escoger la versión "Community Edition", que es gratuita, o si se desea se puede descargar una versión de prueba de 30 días de la versión "Ultimate", que contiene más características.

Cuando se termine la descarga se podrá proceder a la instalación del entorno de desarrollo abriendo el archivo descargado y siguiendo los pasos de la guía de instalación que aparecerá.

Anexo B. Creación de proyectos usando IntelliJ IDEA

Esta parte del anexo resume cómo se efectúa la creación de un proyecto utilizando IntelliJ IDEA y, en particular, un proyecto Android. Este tipo de proyectos son empleados, como se podría suponer, para desarrollar aplicaciones dirigidas a este sistema operativo.

Tras haber instalado el entorno de desarrollo se procederá a realizar la creación de un nuevo proyecto. Para ello, una vez abierto IntelliJ IDEA seleccionaremos la pestaña *File > New Project*, lo que producirá que aparezca una nueva ventana (ver figura 49). En dicha ventana se elegirá el tipo de proyecto que se desea crear, en este caso haremos clic en *Application Module* (seleccionado por defecto) para crear una aplicación Android. A continuación se escogerá el nombre para el proyecto (*Project name*) y la carpeta donde se localizará el mismo. También puede determinarse Después, se avanzará hasta la siguiente pantalla pulsando *Next.*

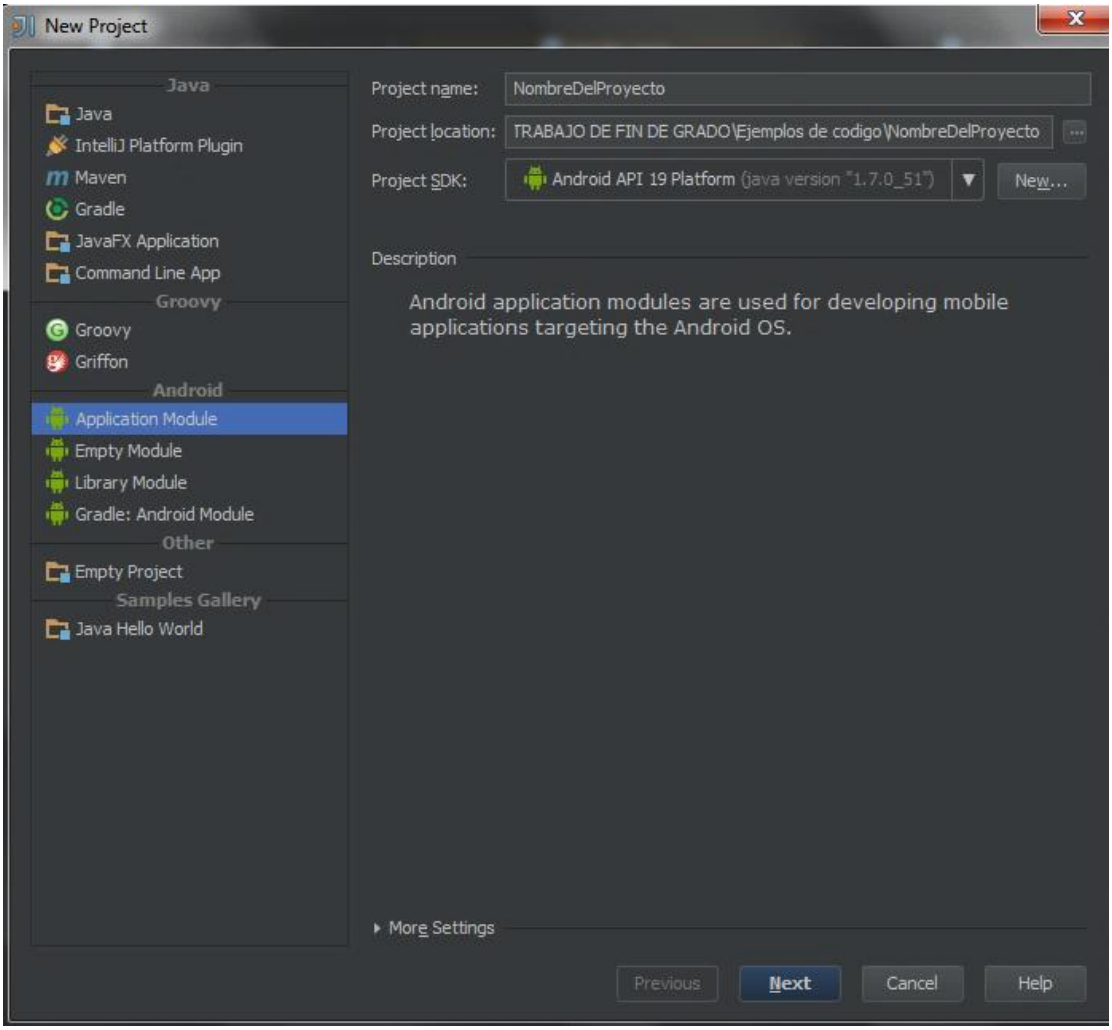

Figura 49. Primera ventana de creación de un nuevo proyecto

En la siguiente ventana encontraremos dos secciones: *Project Properties* y *Target Device*. En la primera se podrá modificar de nuevo el nombre de la aplicación y también se podrá optar por crear la actividad "Hello World!". Se selecciona dicha opción y elegimos un nombre para la actividad en el apartado de *Activity name*, en este caso se puede denominar Pressmatic. En el apartado *Target Device* se determina el dispositivo en el cual se ejecutará o simulará la aplicación[9] al pulsar el botón "Run". Si se desea que el proyecto se ejecute y simule mediante una máquina virtual en el ordenador se elegirá *Emulator* y si se desea que el proyecto sea ejecutado directamente en el dispositivo Android se seleccionará *USB device*[10]. Una vez realizada la selección pulsaremos *Finish*.

[9] Para realizar la simulación de proyectos que lleven asociada una conexión bluetooth como es el caso de PRESSMATIC se requiere el uso del dispositivo Android ya que no se pueden realizar simulaciones de este tipo de conexión con los entornos de desarrollo existentes.

[10] Esta configuración puede ser modificada posteriormente por el usuario accediendo a *Run> Edit Configurations*.

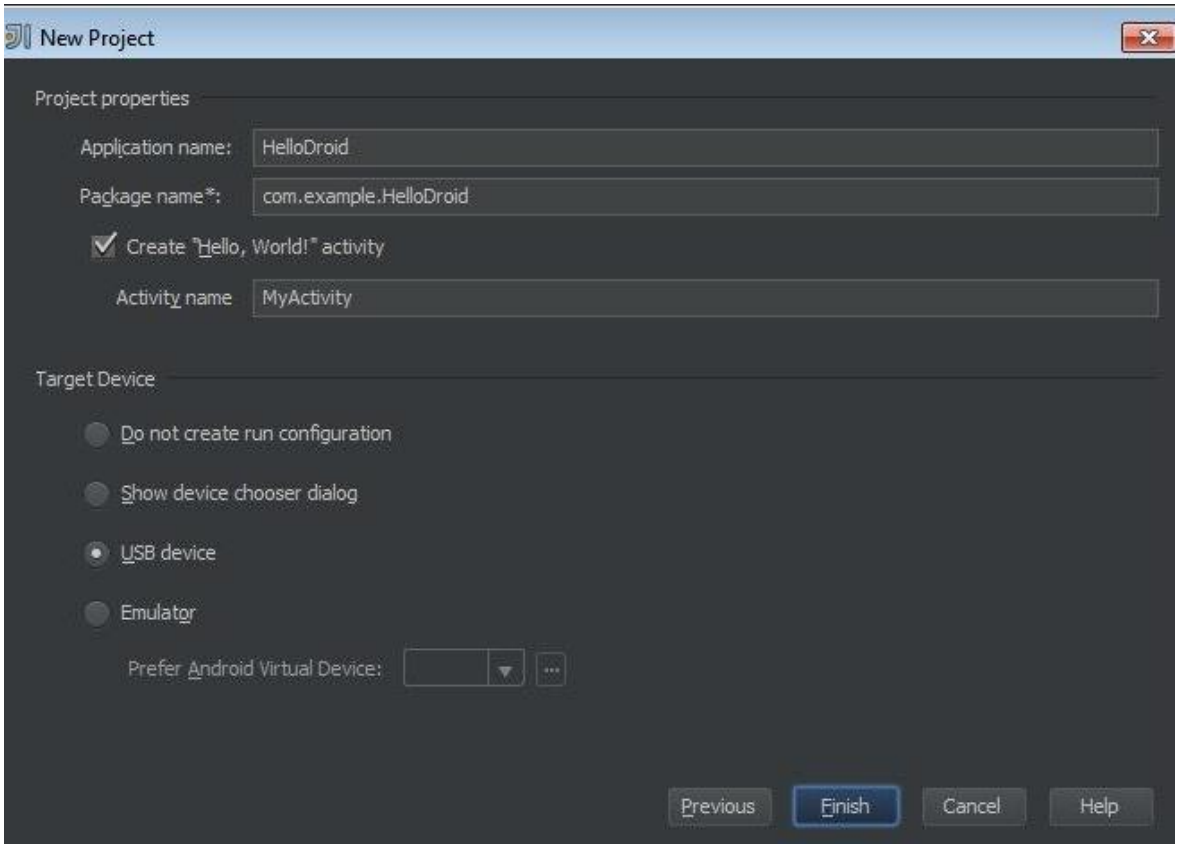

Figura 50. Segunda ventana de creación de un nuevo proyecto

De esta manera el proyecto será generado automáticamente creándose una serie de directorios y archivos que se utilizarán para elaborar las partes de la aplicación y determinar las funcionalidades de la misma, estableciendo la arquitectura de la aplicación.

En el caso de tener cualquier duda o si surgiera cualquier problema, la compañía NetBeans proporciona una guía de iniciación para aquellos que lo necesiten.

Glosario de términos

API	Application Programming Interface
SO	Sistema Operativo
MAC	Media Access Control
sp	Scaled pixel
IDE	Integrated Development Environment

Referencias

[1] G. Barroso de María *"Dispositivo automático de apoyo para uso de herramientas requeridas de movimiento manual de pinzado*". Tesis de máster. Universidad Carlos III de Madrid. Madrid, España. Marzo 2012

[2] A. J. Huete, Proyecto PRESSMATIC, "*II Convocatoria de proyectos inclusivos de Fundación Universia"*, 2013.

[3] A. Rosado García "*Protocolo de comunicación bluetooth y control por voz para PRESSMATIC*" Proyecto de fin de grado. Universidad Carlos III de Madrid. Madrid, España. Septiembre 2014.

[4] R. Martín López "*Programación e integración de interfaz de pantalla táctil accesible para PRESSMATIC*" Proyecto de fin de grado. Universidad Carlos III de Madrid. Madrid, España. Septiembre 2014.

[5] J.M. Fernández "*Tipos de dispositivos móviles*" Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad de Granada. Disponible en: http://leo.ugr.es/J2ME/INTRO/intro_5.htm

[6] Web oficial de la compañía Samsung.
www.samsung.com Último acceso Agosto 2014.

[7] E. Price (18 Febrero, 2012) "*The History of Mobile App Stores*" [Online] Disponible en: http://www.technobuffalo.com/2012/02/18/the-history-of-mobile-app-stores-infographic/

[8] J. Acevedo (27 Agosto, 2013). "*El uso de las aplicaciones en el mundo*". [Online] Disponible en: http://www.pcworldenespanol.com/201308278591/aplicaciones-moviles/asi-es-el-uso-de-las-aplicaciones-moviles-en-el-mundo-infografia.html

[9] Ödm Group (15 Diciembre,2011) "*Living in a mobile world*"
http://odmgrp.com/blog/2011/12/15/infographic-living-in-a-mobile-world/

[10] J. Burgos (27 Mayo, 2013) *"¿Por qué las aplicaciones móviles deben ser accesibles?"*. [Online] Disponible en: http://www.grupofundosa.es/es/%C2%BFpor-qu%C3%A9-las-aplicaciones-m%C3%B3viles-deben-ser-accesibles

[11] Observatorio Accesibilidad TIC Discapnet "*Accesibilidad de Aplicaciones Móviles*" Discapnet. Accesibilidad. [Online] Agosto 2013. Disponible en:
http://www.discapnet.es/Castellano/areastematicas/Accesibilidad/Observatorio_infoaccesibilidad/informesInfoaccesibilidad/Paginas/default.aspx

[12] E. Archanco (20 Mayo, 2014). "*Las 5 apps accesibles e imprescindibles de iOs para utilizar día a día*" [Online] Disponible en: http://www.applesfera.com/aplicaciones-ios-1/las-5-apps-accesibles-e-imprescindibles-de-ios-para-utilizar-dia-a-dia

[13] Web oficial de la compañía Apple.
https://itunes.apple.com/es/ Último acceso Agosto 2014.

[14] Tienda web oficial de Google. Google Play.
https://play.google.com/store/ Último acceso Agosto 2014.

[15] TAPTAP Networks.
http://www.taptapnetworks.com/ Último acceso Mayo 2014.

[16] M. Dubretic (25 Abril, 2014) "*iOs vs Android vs Windows"* [Online] Disponible en: https://www.udemy.com/blog/es/ios-vs-android-vs-windows-la-batalla-de-los-sistemas-operativos-moviles/

[17] Stackoverflow. Comunidad de desarrolladores.
www.stackoverflow.com Último acceso Agosto 2014.

[18] Comunidad de desarrolladores de Android.
http://developer.android.com/ Último acceso Septiembre 2014.

[19] Comunidad de desarrolladores de BlackBerry.
http://docs.blackberry.com/ Último acceso Mayo 2014.

[20] R. Conde (Agosto 2013) "*BlackBerry 10: un sistema operativo diferente, pero cada vez más sólido*"[Online] Disponible en:
http://celulares.about.com/od/BlackBerry/a/Blackberry-10-Un-Sistema-Operativo-Diferente-Pero-Cada-Vez-Mas-Solido.htm

[21] Web oficial de Windows Phone en español.
http://windowsphoneapps.es/ Último acceso Agosto 2014

[22] Code Factory "*Mobile Accesibility*". [Online] Disponible en:
http://codefactoryglobal.com/app-store/mobile-accessibility/ Último acceso Agosto 2014

[23] B.Holton "*Evaluating Mobile Accessibility for Windows Phone*" American Foundation for the Blind. Volumen 15, Número 3. Marzo 2014 [Online] Disponible en:
http://www.afb.org/afbpress/pub.asp?DocID=aw150303

[24] Strategy Analitics. [Online] Disponible en:
http://www.strategyanalytics.com/default.aspx Último acceso Mayo 2014

[25] Web oficial de Android.
www.android.com Último acceso Agosto 2014

[26] Web oficial de la compañía JetBrains.
http://www.jetbrains.com/idea/ Último acceso Agosto 2014

[27] PRNewswire (6 Diciembre, 2012) "*IntelliJ IDEA 12 destapa el lado oscuro de la productividad en programación*" [Online] Disponible en:
http://www.prnewswire.com/news-releases/intellij-idea-12-destapa-el-lado-oscuro-de-la-productividad-en-programacion-182320681.html

[28] J. Tomás Gironés *El gran libro de Android*. 2ª Edición. España. Marcombo S.A. Año 2013.

[29] Android-App-Market.com (17 Febrero 2012) "*Android Architecture – The Key Concepts of Android OS*" [Online] Disponible en: http://www.android-app-market.com/android-architecture.html

[30] "Programación en dispositivos móviles portable" Software de Comunicaciones. Universidad Carlos III de Madrid. [Online]. Contenido disponible en: https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android Ultimo acceso Mayo 2014

[31] Dr. A. Porter. *"Programming Mobile Applications for Android Handheld Systems"* Coursera. Universidad de Maryland. [Curso online] Disponible en:
https://www.coursera.org

[32] CEAPAT. Centro Estatal de Autonomía Personal y Ayudas Técnicas, centro tecnológico dependiente del IMSERSO, Ministerio de Trabajo y Asuntos Sociales. http://www.ceapat.es. Ultimo acceso Agosto 2014.

[33] A. Cereceda *"INVASIÓN ANDROIDE: Un videojuego para Android de inmersión en la realidad. Módulo de geolocalización."* Proyecto de fin de carrera. Universidad Carlos III de Madrid. Madrid, España. Diciembre 2011

[34] Web oficial de Inkscape.
http://www.inkscape.org/es/ Último acceso Mayo 2014.

[35]Proyectacolor *"Círculo cromático"* [Online] Disponible en:
http://www.proyectacolor.cl/teoria-de-los-colores/circulo-cromatico/ Último acceso Agosto 2014.

[36] Aplicación Blueterm.
Código disponible en: http://pymasde.es/blueterm/ Último acceso Mayo 2014

[37] J. Tomás, V. Carbonell. *El gran libro de Android avanzado*. 1ª Edición. España. Marcombo S.A. Año 2014.